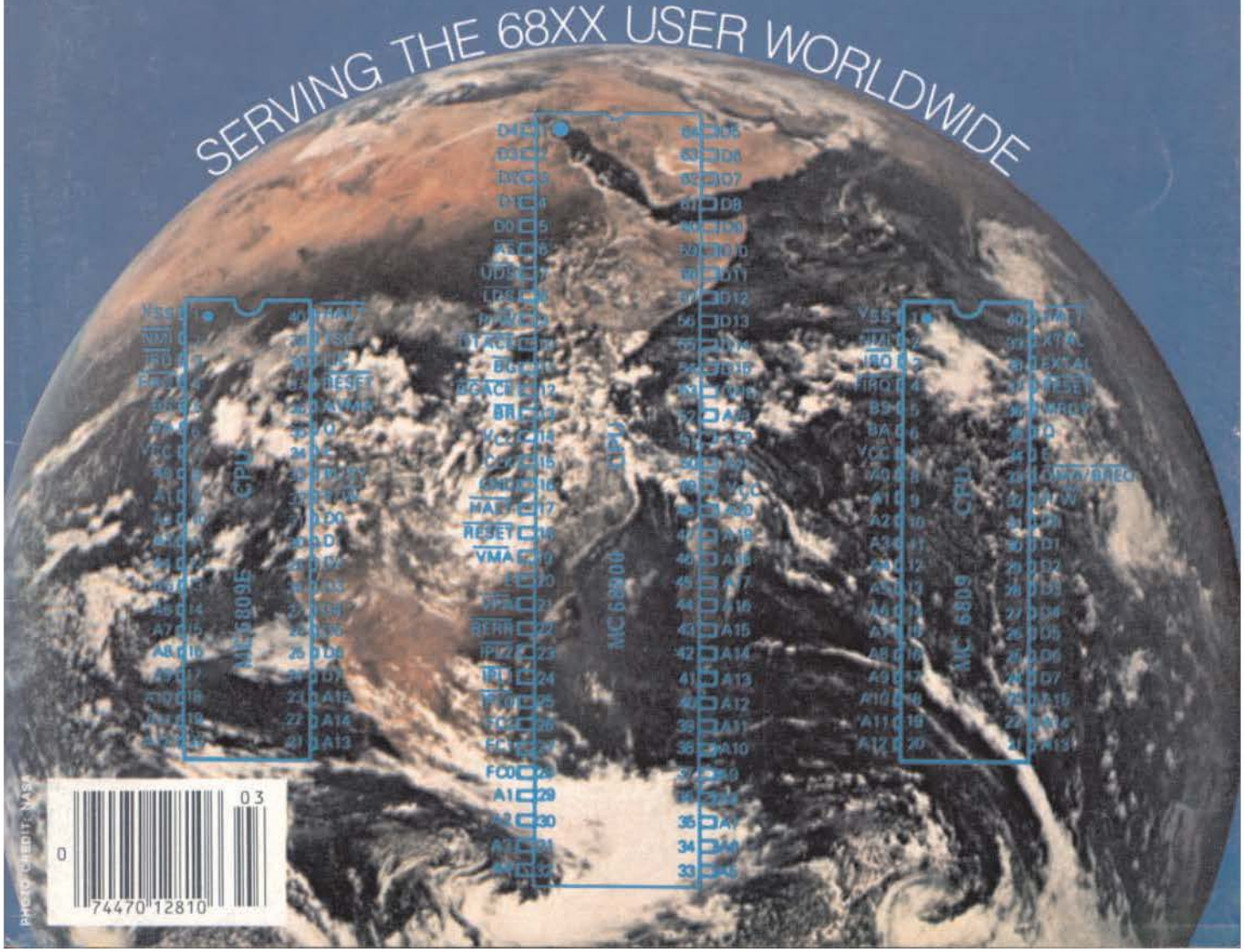
'68' MICRO JOURNAL

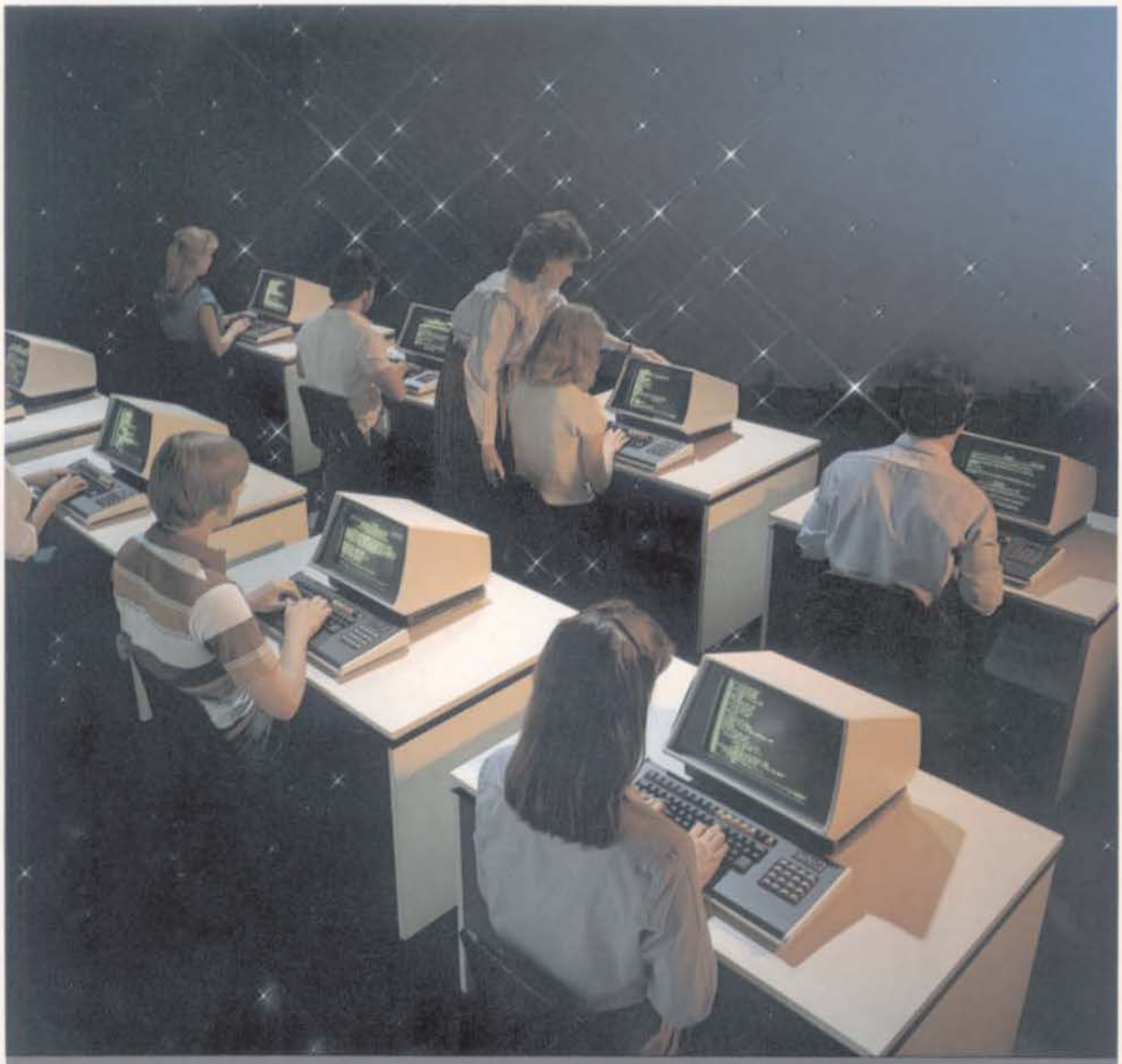
$2.95 USA

Australia	A $ 4.75	New Zealand	NZ $ 6.50
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30:-SEK

VOLUME VI ISSUE III • Devoted to the 68XX User • March 1984

"Small Computers Doing Big Things"





FREE FROM LIMITATIONS, A TEACHER CAN UNLOCK IMAGINATIONS.

By 1985, fully 75% of all jobs will involve computers. So for today's teachers, selecting the right computer is an awesome responsibility. It's a decision you can feel confident about with Southwest Technical Products Corporation's S + computer system. The S + is specially designed for a student's tomorrow and a teacher's today. It's built tough. It's simple to operate. You can teach without becoming a hardware specialist.

The S + is fluent in a variety of languages: BASIC, PASCAL, COBOL, FORTRAN, PILOT, FORTH, "C," and MUMPS and allows you simultaneous use in the classroom. This flexibility enables you to use your own curriculum, not a machine's version. The system also allows for individual learning styles. Students go as fast and as far as the teacher chooses.

Even though it's priced like a smaller personal computer, the S + has the multi-task power and expansion abilities of a professional computer. It can grow with you and your students.

Tomorrow is in your hands, today. Call Fisher Scientific, exclusive distributor of our S + system, and you'll get the key to unlock imaginations.

For more information.
Call toll-free 800-621-4769.
Illinois customers call collect (312) 378-7770.

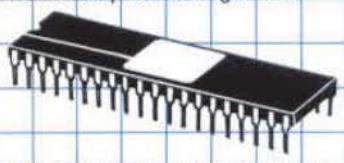
Fisher Scientific
An ALLIED Company

Educational Materials Division
4901 W. LeMoyne St.,
Chicago, Illinois 60651

MICROWARE'S OS-9 IS NUMBER ONE.

OS-9 NOW HAS THE LARGEST USER COMMUNITY

More users now run OS-9 on their 6809 computers than all other operating systems combined. This outstanding success story was no accident — it's due to OS-9's technical excellence backed up by outstanding Microware support. OS-9's Unix-type architecture and totally modular design gives your computer more power and versatility. OS-9 also gives you more possibilities for customization so you can tailor your system exactly to your needs. And aren't flexibility and performance the reasons you chose a 6809 computer to begin with?



OS-9 HAS BEEN CHOSEN BY OVER 50 6809 SYSTEM MANUFACTURERS

OS-9 is now offered as a standard operating system by almost every 6809 system manufacturer, and has been designed into an amazing variety of dedicated systems and products including personal and business computers, process control systems, data and telecommunications systems, and more. In all, over 50 companies and organizations have obtained OS-9 distribution licenses including such well-known names such as General Motors, NASA, Fujitsu, Western Electric, Motorola, Sykes Datatronics, Eastman Kodak, Thomson-CSF, and Tandy Corp.

OS-9 GIVES YOU A SOFTWARE BASE TO BUILD ON

Whatever your application, OS-9 speaks your language! Microware offers BASIC09, an Extended/ Structured Basic, a complete C Compiler, a full ISO Pascal Compiler, the ANSI Standard CIS Cobol Compiler, plus Relocatable Macro Assembler. These high performance programming languages are all fully implemented and deliver unmatched performance and outstanding features. Additionally, OS-9 compatible applications packages such as word processors, screen editors, spreadsheets, business software, and utilities are offered by a rapidly growing number of third-party software houses.

PLUS OUTSTANDING MICROWARE SUPPORT: WE KEEP IN TOUCH WITH YOU

Even when you have the best software and documentation, there can be times when you need questions answered. That's why Microware is committed to giving OS-9 users the best possible **personalized** service. Here are some of the ways we deliver solid support:

- A Software Support Hotline for direct access to our technical staff
- "Pipelines", our free quarterly newsletter
- OS-9 User Seminars, the annual OS-9 community gathering
- a liberal update policy for new releases

Microware does business on a person-to-person basis. When you call you'll find yourself speaking with someone who's both knowledgeable and genuinely interested in helping.

YOU CAN COUNT ON OS-9 NOW AND IN THE FUTURE

Microware is not standing still — we're firmly committed to continuing support for the 6809 and we will continue to introduce exciting new software products for the 6809. We will soon announce OS-9/68000 and programming languages for the 68000 which will be upward compatible with 6809 versions, so if and when you are ready for the 68000 your OS-9 software can go with you.

MICROWARE®

Microware Systems Corporation
Box 4865 • Des Moines, IA 50304
515-279-8844 Telex 910-520-2535

OS-9: BETTER BY DESIGN

'68'

MICRO JOURNAL

Portions of the text for **68 MICRO JOURNAL** was prepared using the following furnished hard/software.

COMPUTERS-HARDWARE
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
S09-5/8 DMF disk-COS1-8212W-Sprint 3 Printer

GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe-OS9-FLEX-Assorted Hardware

EDITORS-WORD PROCESSORS
Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX-Editor-Processor

Great Plains Computer Company, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

CONTENTS

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES
USA
1-Year $24.50 2-Years $42.50 3-Years $64.50
FOREIGN
See Page 60

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors **full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands **ONLY** should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the **.pg** page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

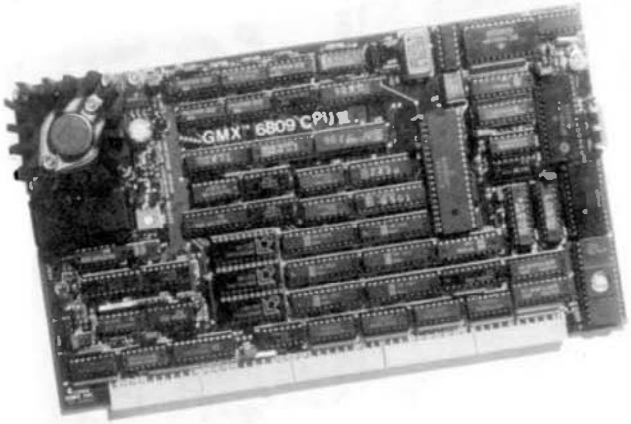
If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

GIMIX STATE OF THE ART 6809 SYSTEMS FOR THE SERIOUS USER.



GIMIX has 19MB or high performance 47MB Winchester Drive Systems and/or Floppy Disk Drive Systems.

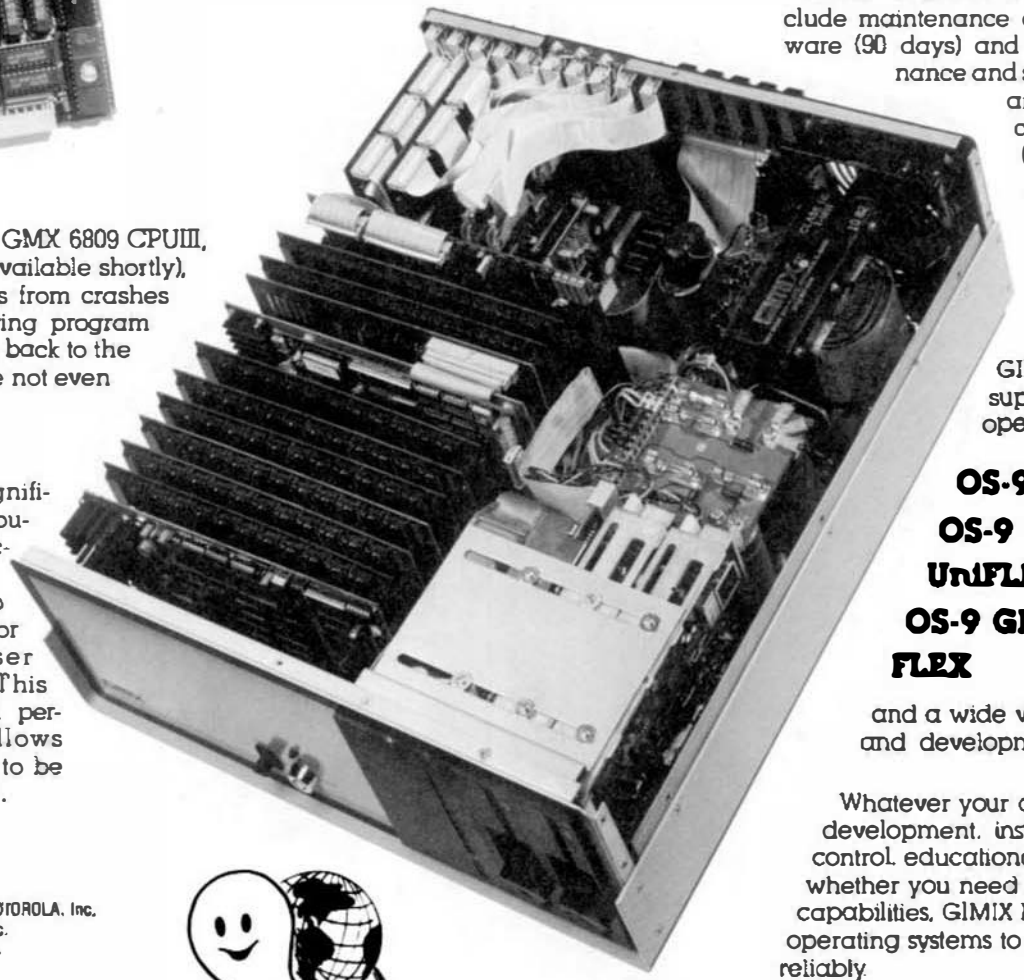
For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer.

GIMIX has single user systems that can run both FLEX and OS-9 or Multi user systems for use with UniFLEX or OS-9.

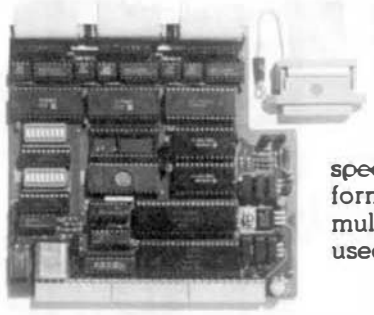
GIMIX versions of OS9 and UniFLEX include maintenance and support by Microware (90 days) and TSC (1 year). Maintenance and support after this period are available at extra cost.

(NOTE: this support and maintenance is only for use with approved GIMIX hardware)

For the ultimate in performance, the Unique GMX 6809 CPUIII, using either OS-9-GMXIII or UniFLEX GMXIII (available shortly), gives protection to the system and other users from crashes caused by defective user programs. e.g. During program development, a programmer who crashes goes back to the shell or the debugger, while the other users are not even aware anything occurred.



The intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions, thereby freeing up the host CPU for running user programs. This speeds up system performance and allows multiple terminals to be used at 19.2K baud.



GIMIX 6809 systems support five predominant operating systems:

OS-9 GMX III,
OS-9 GMX II,
UniFLEX,
OS-9 GMX I,
FLEX

and a wide variety of languages and development software.

Whatever your application: software development, instrumentation, process control, educational, scientific or business; whether you need single or multi-user capabilities, GIMIX has hardware and the operating systems to get the job done reliably.

Please phone or write if you need further information.

GIMIX INC.

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

Microware presents 4 new OS-9 software packages.

1 LEVEL II PRINT SPOOLING SYSTEM

This versatile package gives your OS-9 Level Two System a complete print spooling management capability for time-sharing applications. Features of the spooling system are:

- *Handles up to seven independent spooling devices and queues with "print on first available device" feature.*
- *Prints large block header pages between listings with date, time, user name and job name.*
- *Multiple listing copy option.*
- *Complete forms change capability for each job and device.*
- *Prints formatted or unformatted listings.*
- *Status command displays print queues and status.*
- *User can kill or change priority of queued jobs.*

Available only for OS-9 Level Two Systems.

Suggested List Price: $150.00 Manual Only: $15.00

2 RMA RELOCATABLE MACRO ASSEMBLER

At last — a full feature relocatable macro assembler and linkage editor for OS-9. RMA permits sections of assembly language programs to be independently assembled to "relocatable object files". The linkage editor takes any number of program sections and/or library sections and combines them into a single executable OS-9 memory module. Global data (including indexed and direct addressing modes) and program references are automatically resolved in the process. The macro facility permits commonly used statement sequences to be defined, then used within the program with appropriate parameter substitution. RMA also supports conditional assembly and library source files.

Suggested List Price: $200.00 Manual Only: $20.00

3 OS-9 FILE HANDLER TOOLBOX

Introducing a special toolbox for OS-9 users who do a lot of file manipulation! A collection of 12 useful OS-9 command programs; Most can be used as "filters" using OS-9 pipeline facilities. Included are:

D *— unformatted directory listing with "wild card" matching*
Compress *— does character compression on text files.*
Expand *— restores a "compressed" file to the original state.*
Split *— breaks a file into smaller files.*
Space *— indents lines with optional spacing between lines.*
Code *— decodes any key on a keyboard to hex.*
Qsort *— quick sort for small files, directories, etc.*
Pr *— versatile formatted file printing utility.*
Tr *— transliterates text pattern to substitution pattern.*
Grep *— searches file for a pattern and prints matching lines.*
Xmode *— same "tmode" except changes are made to the device descriptor.*
Count *— counts words, lines, or characters within a text file.*

Suggested List Price $85.00

4 ENTERTAINMENT PACK I

A collection of games and other interesting programs that are not only entertaining but serve as good instructional examples of Basic09 programming techniques. All programs include complete Basic09 source files and can be easily edited to run on standard alphanumeric or graphics terminals.

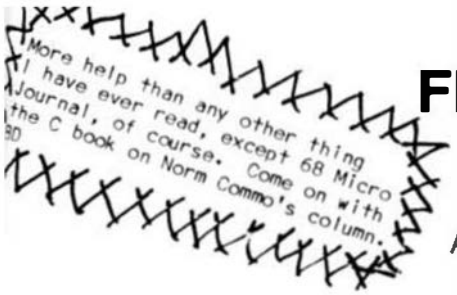
Blkjak *— A Vegas-rules blackjack game.*
Clk *— graphical display of a wall clock on your terminal.*
Dogs *— Greyhound racing with simulated graphics.*
Eliza *— Basic09 version of the famous artificial intelligence simulation of natural language dialogue with a psychiatrist.*
Haiku *— Program that creates original "haiku" prose.*
Quest *— a mini-"Adventure" game.*
Rats *— find your way out of a computer-generated maze — from a rat's point of view.*
Towers *— a graphical display of the solution to the "Tower of Hanoi" puzzle.*

Suggested List Price: $85.00

MICROWARE®

Microware Systems Corporation
P.O. Box 4865 • Des Moines, IA 50304
515-279-8844 • Telex 910-520-2535

OS-9 and Basic09 are trademarks of Microware and Motorola. Unix is a trademark of Bell Labs.



FLEX™ USER NOTES
THE 6800-6809 BOOK

By: Ronald W. Anderson
As published in 68 MICRO JOURNAL™

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES,** in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only — 5" $12.95 — 8" $16.95 plus $2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" $17.95 — 8" $19.95 plus $2.50 shipping and handling.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC
	(And many more)

**Over 30 TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1,.C2, etc. = Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery.

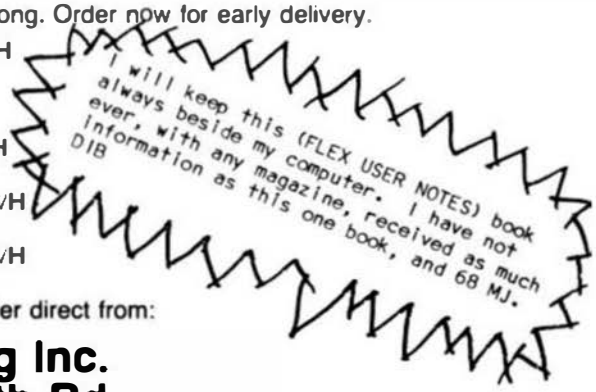
Foreign Orders Add $4.50 S/H

Softcover — Large Format

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H



See your local S50 dealer/bookstore or order direct from:

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

™FLEX is a trademark of Technical Systems Consultants

!!! FREE !!!

Published Monthly by Computer Publishing Inc., Hixson, TN.

$1.95

Bulk Rate
U.S. Postage
PAID
Chattanooga, TN
Permit No. 357

Color Micro Journal

The Color Computer Monthly Magazine $1.95 per issue Vol. 1, Issue 2 October, 1983

THIS 'N THAT

The BIG NEWS this month is that OS-9 has finally arrived for the Color Computer. The ASTOUNDING part of the Radio Shack OS-9 Package, besides the price, is the DOCUMENTATION. You 'Old Time Radio Shack Followers' will not believe what you see. Jon Shirley has been telling us that the main reason for the "lack" of documentation with a lot of their products was the restrictions placed on releasing that information by Microsoft; I

OS-9 on the COLOR COMPUTER

One of the "Operating Systems of the Future" is now available for the "little old Color Computer": OS-9. Freely translated, OS-9 means "Operating System for the 6809" (OS-9 is now being written for the 6809, also). Since it is fairly obvious that UNIX and "UNIX-Type" Operating Systems will be running on just about every computer to come out in the next few years, a whole new language is beginning to appear on the horizon.

Color Computer OS-9; the Package

We had been running a preliminary release of OS-9 on the Color Computer for a few weeks, and received the "Official Radio Shack" version for Review a couple of days ago. To put it mildly, this package is IMPRESSIVE! For $69.95 (Radio Shack Catalog Number 26-3039), you receive a 9 1/2" x 7 5/8" x 2" package containing 4

TMP (Total Management Planning) OS-9 SOFTWARE

THESE OFF-THE-SHELF APPLICATION PACKAGES HAVE CHANGED THE PICTURE FOR PROFESSIONAL SOFTWARE USERS

Here are the comprehensively supported OS-9 application software packages that discerning professionals are selecting today. Each one runs stand-alone or in combination with other TMP packages.

TMP/CALC is a *new-generation* "Electronic Spreadsheet" package that goes a quantum leap beyond ordinary Calc software. Superior speed, and extra features like Dynamic Calculations and Dynamic Overlay, mean you can change a cell *anywhere* on the spreadsheet and get automatic updates wherever they belong. No more "wrap-around" since output to the printer is formatted. "Help" screens prompt without erasing data, and the rows and columns on any worksheet are limited *only* by available memory.
ONLY $250

TMP/Manager is a structured database manager, and is the ultimate driving force behind any total *Management/Marketing Planning* system. Its built-in select/sort/merge module — and optimized machine code — makes it superfast, and provides unusually nimble data manipulation characteristics. Other outstanding features are detailed prompting, easy identification of data fields, and an excellent security system.
ONLY $500

TMP FreeForm fills the void between a structured DBMS and word-processing: it's an "electronic index card" package with an endless list of valuable applications such as parts cataloging, look-up inventories and listings of *any* kind. Type anything on up to 32,000 electronic "cards" of up to 9 pages each that reside in "file drawers" (databases). Up to 13 key-words can call each page.
ONLY $150

TMP/Front-End integrates any or all of the packages described above, and integrates TMP with word-processors, BASIC and other high-level languages.
NO CHARGE with TMP/Manager

TMP T.M.

TMP PACKAGES ARE EXCEPTIONALLY FRIENDLY. EACH COMES WITH MULTI-MEDIA TRAINING AND OUTSTANDING DOCUMENTATION

AUDIO CASSETTE TRAINING TAPES are provided with each TMP software package. These crisp, professionally produced training aids take a new user through "hands-on" exercises that instantly build confidence, and minimize reference to manuals.

CLEAR DOCUMENTATION is specially prepared for each TMP package. Remarkably readable because they are conversational in style and logically organized, this series of manuals *further* assures quick user comfort and productivity with TMP software. Includes sections on how to best utilize the system; integrating user-written software with TMP packages, and other bonus topics.

VIDEOTAPE PROGRAMS are available to TMP Dealers. They dramatically take the viewer into real business environments for a close look at how a TMP software package is utilized. Video programs can be personalized for specific organizations. Programs may be specified for most popular tape formats.

THE BOTTOM-LINE IS THIS: Software is only as effective as its training and documentation. TMP packages are *incomparable* in both areas.

RUN TMP IN A SMOKE SIGNAL CHIEFTAIN™ COMPUTER TO EXPERIENCE THE KIND OF SPEED AND POWER YOU CAN EASILY AFFORD *TODAY*

Endurance-Certified Chieftains are consistently among the fastest computers in CPU and I/O speeds according to the widely respected BENCHMARK REPORTS. That, combined with almost legendary reliability, makes Chieftains the logical choice in computers using OS-9 operating systems.

Configurations range from floppy-disk systems to multi-user, multi-tasking Winchester hard disk systems with tape back-up; performance approaching mainframes at prices you can live with.

TMP software packages for most OS-9 systems and Chieftain computers are offered exclusively from Smoke Signal. Inquiries from non-Smoke Signal dealers are invited. Call (213) 889-9340.

SMOKE SIGNAL
Chieftain Computers
31336 Via Colinas • Westlake Village, CA 91362
(213) 889-9340

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

MORE ON ASSEMBLER VS COMPILER

I had just mailed my last column with the response from Dan Farnsworth, when a letter arrived from my "printer pal" Art Weller of El Paso, TX. Art writes interestingly, and I think I will just quote him here.

"I'd like to add my comment to your 'argument' with Dan Farnsworth and I'll do it separately so that, if you want to, you can put it with the pile I'm sure you'll get from others. Mostly, I guess, you'll hear from programmers who'll say, 'Yeah!, yeah! -- maintain a library of assembler routines'. Or, 'You mean some people still use assembler?' Well, I don't think I have enough experience as a 'programmer' to get involved in that aspect, but I do have a lot of experience as a 'manager' with exposure to computers and programmers. There are several points I tried to make with limited success at the time. Maybe I can do better now that I have some 'hands on' experience and speak a little of the language.

"You didn't have to specifically say that the discussion has to do with productivity -- it's obvious. That will get a manager's attention every time and should, in fact, concern even a free lance programmer (isn't he his own 'management'?). To make it a little easier to get to the point, let's assume a situation in which the programming, computer operation, and 'user' are all 'in-house' (as it was in my case). Variations of this, as for example, production of software for sale (user=customer) don't appreciably change the problem.

"I can't think of any activity that is more 'labor intensive' than computer programming; can you? It consists of just about 100% labor and you can't even hold the product in your hand to examine it. Worse yet, it's usually very expensive labor (relative to the rest of the workforce) and these costs must be recovered or amortized somewhere in the organization. That's just a gobble-de-gook way of saying that programming effort must pay for itself by reducing costs elsewhere in the company. More efficient use of the computer, a reduction of workload for a computer user/operator, or more effective use of some other resource.

"That's really what I wanted to say, but it's a point that seems hard to get across. Perhaps a few hypothetical examples would help. If an application is I/O bound (to pick an obvious one), what matter if a few milliseconds of machine time can be picked up with a little re-programming? How long would it take to amortize the cost of the changes? Here's one of my favorites. How about making the software nice and compact, using an absolute minimum of memory space? Well, if that also means truncated, cryptic prompts and requires more extensive training of each person who uses it (and re-training for infrequent users!), then some potential savings won't be realized. And it should be emphasized that these are recurring costs that go on forever, or as long as the program is in use.

"As I see it, this is germane to the argument since the choice of language will have a great deal to do with the 'efficiency' of the programmer. The choice will also be somewhat peculiar to the individual programmer as a reflection of his skills. However, it is also true that some applications are very inappropriate for some languages for they do not provide the features required by the application. While some languages allow for the development of the missing functions; that, in my judgement, is a mis-use of the programmers time. The optimum arrangement is to select a language that is ideal for the specific application and a programmer who is skilled in its use and has prior knowledge of the application.

"May I take sides at this point? Dan is quite right about the desireability of maintaining a 'library' of previously programmed and debugged routines that can be re-used in later applications at a great savings in time. But that is also true of the 'higher' languages and therefore I don't feel that it is a valid argument for selection of any language over another."

Well, there you have one reader's opinions, from the point of view of productivity and profitability. I'll let Art's letter speak for itself. One comment though is that Art was wrong about my mail being heavy with responses and opinions with regard to this discussion. I've received only a few other letters at this point. James Kuzdrall wrote the following paragraph in a letter to me.

"I was just reading your November column, and I have been using the same scheme of assembly language programming as Dan Farnsworth. I have had to abandon it in recent years, however, because often now the 6809 or 6800 will not be the target system. In fact, often several chips must be accommodated. Like so many others, I have turned to C as a "universal assembly language". The method carries over, however, as I put more and more of my favorites into MYLIB.LIB. Now if I could get good cross compilers, I would be all set. I hate to have to borrow or rent a system just to compile a program on it."

As I was writing this, the phone rang. It was Frank Hoffman, author of CRASMB and other good software. Among other things, he mentioned to me that he had read the discussion on Assembler vs Compilers, and he commented that a Compiler is another tool of which we computer users can avail ourselves. We can either write in assembler and keep a lot of detail in our heads, or let a compiler take care of some of the detail for us.

COMMENTS ON "C"

Some time ago I received a letter from James McCosh containing some good information about "C", and I will quote it here. I had said something about a friend calling "C" a universal assembler.

"I totally agree with you about C. It is not the best or most elegant language ever invented, but provided not too much is expected of it, it provides an efficient solution to a very large class of problems. It can be described as a 'high-level assembly language' or a 'medium-level' language. I was an ardent supporter of assembly code before meeting C. This was mainly because I liked the freedom and flexibility coupled with the extreme efficiency possible. With C, you lose some efficiency with virtually no loss of the other two attributes and gain two enormous bonuses: programs take about a tenth of the time to get to a professional standard and they are then portable. However, C is not for beginners and, perhaps should not be given to anyone who has not had a thorough grounding in Pascal first. I really only got the best out of the language when I had a good grasp of structure and more esoteric things such as recursion

and coroutines."

I had mentioned that I am impressed that all the "small" C implementations are true subsets of the larger implementations. James responds with:

"The reason that subsets of C are just that is that it is essentially a very 'small' language. Just look at the list of reserved words. Also, quite a large part of the language is redundant in the sense that one can get by with a small subset and build the rest from that. The power and portability comes from the idea of a library, which can itself be written in C, and which can be adapted to any environment.

"Another reason for its increasing popularity is its essential practicality. It was designed by people with tremendous theoretical knowledge but who required an efficient language to write an operating system. The result is sometimes a little weird and gives the newcomer a few problems and the computer scientist nightmares. Unlike Pascal, the definition of C was written down by looking at the way the compiler behaved after it had been in regular use for years! The philosophy behind it is to give the programmer a means of writing good structured code and facilities to access the operating system and machine directly. Almost no help is given in preventing the programmer from doing stupid things so one has to have his wits about him..."

In my communications with Introl, I received an interesting letter from their Paul Voith. I had mentioned the very large size of the library functions scanf() and printf(). Paul responded very thoughtfully to my questions. Taking printf() as an example, Paul pointed out that because printf() is a library function, it must be capable of handling any data type, as signaled by the control string. Considering the fact that printf() is written in C, it really is a feat to be able to code it at all. Pascal has its WRITE() function, but it is not a library function, and the type of the data passed to it is known at COMPILE time. The printf() function in C 'finds out' the type of the data being passed to it at RUNTIME. printf() must in fact interpret the control string to determine how to handle the printing of the variables. This distinction is really more responsible for the large code than the fact that printf() is written in C. I hope I've put Paul's thoughts into words that are clear enough so you could follow. Of course the same remarks apply to the scanf() function.

I've found that it is not necessary to include these high level functions in a small program. Many times the only output will be a string or an integer. You can write an outstr() (for out string) function or a printi() (for print integer) to be rather simple since it only has to do one thing. The only inconvenience is, of course, that you can't mix a string and an integer in a single print statement.

GRAPHICS

I've recently gotten into some bit graphics on a printer. Of course, as soon as I finished the project for a new Epson, we decided to switch printers and do similar graphics on a DEC LA-100. First I'd like to say that I realize that DEC has been around for a long time, and that historically, they had 12 bit minicomputers a long time ago. DEC always used octal representation of their codes for that reason. However, octal makes little sense with 8/16 bit data, since dividing a 16 bit value into two 8 bit values doesn't divide it at an octal digit and the values change. For example the two 8 bit octal values 007 and 145 combine (007 high order) to form the 16 bit value 003545. Anyway, the codes given in the manual as each control code was explained, was in octal and ASCII. It didn't make the job impossible but it would certainly have been handier to have decimal or hex values given also.

I had fortunately written the graphics program modularly, and once I had all the control codes operating, it was a simple matter to change the mode setting procedures to the new codes. Of course there were several other problems. The DEC had no graphics mode density equivalent to the Epson, and I had to change the number of dots in the graphics to get the same width graphics printout. Epson uses 8 print wires and selects them by simple binary code, top wire being highest order bit. DEC uses only 6 wires, and the highest order bit is at the bottom. In addition, DEC uses the simple binary code with an offset of $3F, which must be added to each calculated bit pattern. Actually the differences sound more complicated than they were, and it took just one day to get the printer connected to the development system through a suitable cable, set up its dip switches for suitable initial modes of operation, dig out all the needed control codes, and modify the software so that it would work with this printer. Incidentally, the DEC LA-100 has nice letter quality printing, and the possibility of using different type fonts available on plug-in ROM cartridges, and software selectable.

I'd like to get into the graphics program for an arbitrary function, but space doesn't permit it this time. Perhaps next month I can present such a program. I developed a way to reduce the array size required to output a graph, and I will go into the technique when I describe the program. Essentially I printed 30 lines of 450 graphics characters, about 13.5K characters, using only 1350 memory locations for the job. The trick is to treat each character as a blank, OR in the graph grid lines when the character is output, and only store the graphics characters for the calculated points that are to be printed. If I get the time, I'll work up the program in Pascal. I've done it in PL9 and BASIC, at this point. The technique can be extended easily to plot multiple curves on one graph at the expense of more memory to hold the plot data.

Don Williams In Hospital

As I write this, Don Williams is in the hospital recovering from bypass surgery. The news so far has been good, and I'd like to wish Don a speedy recovery. By the time this is published, he will be back to full strength, and probably feeling better than he has for a long time.

New from Motorola

Lest we all think Motorola has dropped the 8 bit processor efforts in favor of the 16, I thought I should include this information. It was supplied to me courtesy of Motorola Semiconductor Products, by Woody Baker. This information was new and hot off the press when I received it. Unfortunately, I was quite busy at the time, and this column runs three months in lead time, so that by the time you see this, it won't be new news.

As you know, Motorola has introduced a line of special 8 bit processors for industrial and commercial applications, (such things as Microwave oven controllers, washing machine controllers, vending machine controllers, etc.) Among these are the 6801 and 6805, each with a number of different configurations. Well, they have just released the information on the 68HC11A4. The best way to tell you about it in a few words is to quote their general description paragraphs.

"The MC68HC11A4 is a single-chip microcomputer that utilizes HCMOS techniques to provide the low-power characteristics and high noise immunity of CMOS plus the high speed operation of HMOS. On chip memory systems include a 4K byte ROM, 512 bytes of electrically erasable programmable ROM (EEPROM), and 256 bytes of static RAM. The MC68HC11A4 microcomputer also privides highly sophisticated, on-chip peripheral functions including: an 8-channel analog to digital converter, a serial communications interface (SCI) subsystem, and a serial peripheral interface (SPI) subsystem.

"New disign techniques are used to provide a 2 MHZ nominal bus rate. The timer system is expanded to provide three input capture lines, five output compare lines, and a real time interrupt circuit. This gives the MC68HC11A4 one of the most comprehensive timer systems found on a single chip microcomputer. Other features of the MC68HC11A4 include: a pulse accumulator which can be used to count external events (event counting mode) or measure an external period (input gates accumulation of internal clock - E/64); a computer operating properly (COP) watchdog system which helps protect against software failures; a programmable clock monitor system which causes generation of a system reset in case the clock is lost or running too slow; and an illegal opcode detection circuit which provides an unmaskable interrupt if an illegal opcode fetch is detected."

The data sheet goes on to describe the operating modes, the main two being the single chip mode, and the multiplexed mode. In the first, the processor uses its pins to provide both parallel and serial I/O. In the second, some of the parallel ports are used for data and address functions. In that mode, the 6811 can address a full 64K of external memory, though it utilizes multiplexing of the address, so that 8 of the address bits are output at one time, and therefore 8 must be latched by external circuits.

The most interesting aspect of the 6811 is the implementation of an expanded instruction set. The 6811 has X and Y index registers, but no U (user stack) register. It does not have indexing via the S (system stack) pointer. The new instructions that may be of considerable use to an assembler programmer include ABY (add B to Y), ASLD (arithmetic shift left D), the 6800 instructions CLC and CLI (clear carry and clear interrupt mask), CPD (compare D), LSLD, LSRD, PSHX, PSHY, the 6800 instructions TAB, TAP, TBA, TPA, TSX, TXS, the Y register counterparts TSY, and TYS, and the special XGDX and XGDY (exchange or swap D-X or D-Y).

The best, I've saved till last. The 6811 has the MUL instruction of the 6809 but it also has two DIV instructions. IDIV is an integer divide (i.e. 5/3 = 1). There is also a FDIV or fractional divide, in which the contents of D are divided by the contents of X. In the IDIV case, the quotient is returned in X and the remainder in D. The data sheet is a little unclear on the FDIV operation, but I assume that the quotient end up in X and the fractional part of the quotient ends up in D. That is, the quotient is 32 bits long, and has an assumed binary point between the X register contents and the D register contents.

There is also another new instruction class called BCLR and BSET. These allow setting or clearing bits in memory via direct or indexed addressing modes. For example BCLR 3,X %11110000 would clear the four high order bits at memory location 3,X. That single instruction would therefore replace the sequence of three instructions: LDAA 3,X ANDA #%00001111 STAA 3,X. BSET 3,X %11110000 would set the four high order memory bits at that location.

There are also two other instructions BRCLR, and BRSET. BRCLR 7,X %01000000 LABEL, would branch (relative) to the label "LABEL" if B6 of the byte at address 7,X were clear. BRSET would of course branch if the indicated bit or bits were set (all of the bits have to be 1's in the case of testing multiple bits).

Thus the 6811 has some very "powerful" new instructions that will make it quite nice for control applications. Woody Baker indicated to me that Motorola will be publishing a set of macros that will duplicate the action of the new instructions when included in assembler code for a 6809. Potential users of the new chip can therefore begin developing assembler software for it.

The 6811 has Direct, Extended, and Indexed addressing. Of course the instructions that refer to registers use inherent addressing, and the branch instructions use relative addressing. Indexed addressing uses an 8 bit UNSIGNED offset, and relative addressing uses an 8 bit SIGNED offset. Motorola indicates that the instruction set is "basically a proper extension of the MC6801 CPU. In addition to its ability to execute all M6800 and M6801 instructions, the MC68HC11A4 CPU has a paged operation code (opcode) map with a total of 91 new opcodes. Major functional additions include a second 16-bit index register (Y register), two types of 16 by 16 divide instructions, a STOP instruction, and bit manipulation instructions." The data indicates that the STOP instruction will stop the internal clocks. It is not clear how the processor is again started after a STOP.

The most encouraging thing to me about the introduction of the 6811 is the fact that Motorola has not lost sight of the major applications of microprocessors outside of the area of small computer systems (the kind with a terminal, disk drives, and printer attached). They have also considered "portable" or battery operated applications by having designed this chip in HCMOS so that battery operation is feasible for such applications.

I suppose that most of the applications for this chip will be in the area of controls. Wouldn't it be nice if Motorola were to introduce a "super 6809"? (Would they call it the 6819)? Having these new instructions in addition to all the 6809 instructions would allow compliers to genrate code that would run considerably faster. Try a 6809 benchmark for integer multiplication using the MUL instruction vs a divide benchmark using a standard "shift and subtract" algorithm. It should be very clear that the divide operation slows the 6809 considerably. Just think how the banchmarks would look with a built-in divide instruction!

Thank you Woody for the advance copy of the data sheet and programming summary sheet.

OS9 USER NOTES

By: Peter Dibble
517 Goler House
Rochester, NY 14620

The CoCo

I now have a Radio Shack Color Computer with OS-9. I had hoped that this column would be about my first experiences as a new CoCo/OS-9 Level One user, but I have only had a few hours to play with the new machine and this column is due.

Even just a few hours with the CoCo version of OS-9 is enough to form some first impressions. First, that really is OS-9 in there. All the standard commands and utility programs are included. Even XMODE, which didn't come with my Level Two system, was on the CoCo OS-9 disk. I am impressed with the performance of the CoCo. I am used to a two megahertz GIMIX system, and the CoCo is distinctly slower than that; but, I bet Basic09 on a CoCo would give an IBM-PC running its version of Basic a good race. I hope I have a chance to do some benchmarks soon.

For a user moving from Color Basic to OS-9 the change must be wonderful, but confusing. OS-9 brings out much of the power hidden in that little off-white box. It also demonstrates the limitations of the Color Computer. After this column I intend to concentrate on positive aspects of the CoCo, but right up front I have to say that my new CoCo is a sit-down lawnmower with the soul of a Grand Pre racer. I want to get my complaining out of the way early, so this column is elected.

On the hardware side, I guess my complaints can be summarized as: this computer seems to have been designed to sell for under a thousand dollars. It is really unfair for me to think that this computer should have DMA (Direct Memory Access) for its disk I/O and a chip to do its serial I/O. By doing those tasks in software Radio Shack hurt OS-9's performance, but they also kept the cost of the computer down.

Certainly, my main reaction to the Radio Shack version of OS-9 was pleasure, but that didn't keep me from finding a few things to complain about. In my last column I hinted that the disk driver included with CoCo OS-9 doesn't adhere to OS-9 standards. I didn't make a strong statement because I didn't know from personal experience. I can tentatively confirm the information now -- the CCDisk disk driver doesn't seem to refer to the parameters set in the disk device descriptors.

The documentation that came with OS-9 was also a disappointment. I expected entirely new books explaining the trickier aspects of OS-9 so any fool could understand it. The manuals I got are just prettied-up versions of the Microware manuals with some parts missing. The documentation seems to have been very quickly done. I checked out the section on device descriptors first thing; the manual includes a full description of the device descriptor with no indication that some parameters don't work on the CoCo. Most of the information from Microware's manuals about adapting OS-9 to a new system are missing from Radio Shack's OS-9 documentation.

My complaints may sound significant, but they are not. The hardware limitations of the Color Computer are no worse than one would expect in a low-cost computer. The limited disk driver is only waiting to be replaced by a more general one. If no one else writes one, I may do it myself. The documentation problem is an invitation to people like me. If OS-9 on the CoCo continues to be as big a success as it has been, books will appear about it in fairly short order.

Notes on Compuserve

I spent over two hours reading through the messages in the new OS-9 SIG on Compuserve. That bulletin board is really picking up! People are beginning to buy Basic09 for the CoCo and are having trouble installing it. Some messages went something like: I installed Basic09 on my system and it doesn't work -- HELP. I can't imagine how anyone is able to figure out what went wrong from that kind of complaint; I certainly couldn't. Several other people gave more detailed descriptions of their troubles. It sounded to me like they were having troubles with directories.

When you start OS-9 running it will find a directory called /D0/CMDS on your system disk. This is the directory OS-9 will always execute programs out of unless you explicitly direct it to another directory. Specifically, if you give the command

BASIC09 OS-9 will look for an executable file called BASIC09 in the /D0/CMDS directory. If it finds the program, everything is fine; otherwise, OS-9 will search the default data directory (initially /D0) for a file called BASIC09. If BASIC09 is found in the data directory it will be taken as a shell command file, and a shell will be started up to execute the commands. If that file turns out to be full of the machine code for Basic09, the shell will be understandably confused. If you copy Basic09 from its distribution disk to the root directory for your system disk (which is what the command:

copy /D1/basic09 Basic09 will do) your shell will get wrapped around the axle in about the way I just described. The way to avoid that problem is to put Basic09 in your execution directory with a command like:

copy /D1/basic09 /D0/CMDS/Basic09

The system disk on my CoCo is very full. If I had any number of my own programs on that disk it would overflow. When that happens it is time to divide the files on that disk between two disks. One way to split things up is to put Basic09 and a few other programs that are frequently used with Basic09 on a disk by themselves, and replace the system disk with the special Basic09 disk when it is time to use Basic. There is nothing wrong with the idea, but there is a nice pitfall waiting here too. Directories are files, and, to save time, OS-9 remembers where the files you are using are on disk. When you boot OS-9 it determines where the directory /D0/CMDS is and will look right there next time it needs to find a program. If you pull out the system disk and put in your special Basic09 disk, OS-9 will read the location on the Basic09 disk where the /D0/CMDS directory was on the system disk. In the best case you will get a meaningful error, but you may not. The way to get around this problem is to remember to change your execution (and perhaps your data) directory when you change the disk it is on. That is:

Take the system disk out
Put the Basic09 disk in
type CHX /D0/CMDS which will cause OS-9 to find the /D0/CMDS directory again. Of course, if you decide to call the execution directory on your Basic disk something other than CMDS, that's fine; just change the execution directory appropriately. For example:

OS9: CHX /DO/BASIC.CMDS

If you put Basic09 on a disk separate from many of your other programs you may find yourself unable to get at some important program while you are using Basic09. There are at least three ways to solve this problem.

OS-9 lets you load programs into memory and keep them there. You don't want to load too many because main memory is a very limited resource, but sometimes it can prove very useful to have a program or two in memory. If you insert your Basic disk, load /DO/CMDS/basic09 (note that I specified the full directory name instead of changing the execution directory -- either way will work, but this way I won't need to change the directory back), then remove the Basic disk and put the system disk back in. Now Basic09 is in main memory. You can see Basic09 in the output of the MDIR command, and the MFREE command will show that there is much less free memory in the system than there was before you loaded Basic09. Now, if you type

OS9: basic09 you will find yourself in basic much faster than when it had to be loaded from disk. To get rid of the copy of Basic09 in main memory use the UNLINK command:

OS9: UNLINK basic09

If there is some small number of small programs you want to use from within Basic09 you can load them into memory while the system disk is mounted. For example:

OS9: LOAD copy
OS9: LOAD list remove the system disk insert the basic disk
OS9: CHX /DO/CMDS and perhaps change the data directory
OS9: CHD /DO/BASIC.PROGS then start basic09
OS9: BASIC09

If, for one reason or another, neither of these tricks will serve, you can change the execution directory from within Basic09. For example, starting from a time when Basic09 is running with the basic disk on drive /D0:

replace the basic disk with the disk with the programs you need
B: chx /DO/CMDS - or whatever do what needs to be done, then, before exiting from basic, replace the basic disk in the drive.

The Basic09 CHX command only changes the execution directory within Basic09 and any programs that are run from it. When you exit from Basic09 the directories that were active before you started Basic09 will be active again.

Thank You GIMIX

Ever since the CoCo version of OS-9 was announced with a different disk format from all other versions of OS-9 the users of large OS-9 systems have been grumbling about the incompatibility of our disk formats and the CoCo format. GIMIX has released a new floppy disk driver for their systems that supports reading and (if you have a 40 track drive) writing disks in CoCo OS-9 format. I am very grateful, and I am sure I represent many other OS-9 users when I thank GIMIX for their efforts.

A Handy Shortcut

I always use 32K when I run Dynastar, and I almost always use 24K for the Microware Assembler. I am seldom content to use the minimum memory requirement given in the module header for any program. I have modified the module headers of several programs so they will automatically request the amount of memory I usually request for them. Debug can be used to do this. The commands which will modify Dynastar (DS) to default to its maximum memory size (32K) instead of the minimum (8K) are:

```
load ds
debug
I ds
.   .+b      To point at the permanent storage size in
             the module header.
             The value of this byte is $20
=7F
=FF
Q            The change is made so quit debug
```

Test ds to make certain the new default is working.

I first made certain I could edit a large file, then

Invoked procs from within ds and noted that ds was

using 128 pages. If you want to make the change permanent use the following sequence:

```
OS9: save /DO/temp ds
OS9: verify U </DO/temp >/DO/CMDS/ds2 Check its attributes
OS9: attr /DO/CMDS/ds2 You will find that the execute and public execute attributes are missing, so turn them on
OS9: attr /DO/CMDS/ds2 e pe Save the old version
OS9: rename /DO/CMDS/ds old.ds Install the new one
OS9: rename /DO/CMDS/ds2 ds
```

"C" User Notes

Norm Cosmo
3 Pryor Road
Natick, MA 01760

This month we have a pot-pourri subjects to cover. I have received some floating point packages from two different 68 Micro readers who are in the consulting business. Both of the packages are designed to work with the Introl compiler and run time package. I have also had a chance to read "THE C PUZZLE BOOK" and will review it here; and finally, I have a more esoteric example of using structures in C to write command dispatch tables.

9511A FLOAT PACKAGE

A couple of months ago I received a package from James Kuzdrall called Float-C. The following description is taken from the manual.

"Float-C is a set of C library modules which provide an efficient and simple interface between the Am9511A Floating Point Arithmetic Processor chip and the Introl C compiler. All scientific functions and arithmetic operations conform to the proposed IEEE standard covering a range of 2e+38 to 2e-38 rather than the restricted range of the 9511A, 1e+19 to 1e-19."

I do not have a 9511 processor on my system so I can't review this product directly. I can just pass on my impressions based on the quality of the documentation and some phone conversations with the author.

The package that I received is for the FLEX version of the Introl compiler and looks very interesting. It provides a number of useful scientific function, such as:

```
sqrt()  sin()  cos()  tan()  asin()
acos()  atan()  log()  ln()  exp()
```

There are also many lower level function for converting between the IEEE floating point format and the format required by the 9512A and manipulation the 9512A stack. I won't go into them here since most programmers will be quite content just using the functions mentioned above.

Of course, you must have a properly installed 9511A chip in your system.

The package comes with utilities that help generate a new STDLIB.LIB by changing the old float routines to the new 9511A routines and by altering $RUN, $RUNLINE and $RUNFULL. This also includes the regular arithmetic operations of addition, subraction, mutiplication and division.

The documentation appears well done and delves deeply into some of the problems created by the difference between the IEEE and 9511 floating point formats and how these problems were overcome. It also describes how the package handles rounding, errors (via the Introl Nan feature), accuracy limits and how and where errors can slip in.

How well does it run? Again, I can only go by what the manual said but here is a sample of how it ran against other combinations for the "float.c" program published on page 92 of the August, 1982 issue of Byte

complier	cpu	math	time	note
Digital Research	8088(1)	8087(1)	11.8	(5)
Quantum	8088(2)	8087(2)	13.0	(5)
Introl, FLoat-C	6809(3)	9511A(4)	15.3	
DeSmet	8088(1)	8087(1)	15.5	(5)

Notes: (1) 8088 and 8087 clocks were 5Mhz
(2) 8088 and 8087 clocks were 4.77Mhz
(3) 6809 clock was 2Mhz
(4) 9511A clock was 4Mhz
(5) double.

While this doesn't really compare apples against apples (the 8087 times were for doubles after all), it does show that there's a lot of meat to the 6809/9511A combination for floating point work. Since I couldn't test it here I won't spend much more time on it, but it seems like a worthwhile package. If you want more info mation you can write to:

James A. Kuzdrall
Introl Service Company
Box 127
Nashua, NH 03061

tel: (603) 883-4815

ELEMENTARY MATHEMATICAL FUNCTION PACKAGE

This package is a library of scientific functions for the Introl complier. It consists of a diskette containing a library of functions, and a manual. The author, Delbert Franz, had talked with me some months ago about the suitability of the C language for his line of business. I'm glad he decided to use C (on a CoCo) since this package is a fruit of that decision. I have a preliminary release and therefore don't have complete sales information. If you have an immediate need, I suggest that you contact him at:

Delbert D. Franz
752 Ormonde Drive
Mountain View, CA
94043

I haven't had a chance to try the package out yet, but I want to get the word out to you readers since there may be more than a few of you who need a package such as this. I will give a brief description of the package based on reading the manual.

All the function are floats (single prescision) since Introl does not presently support type double. They are based on the book "Software Manual for the Elementary Functions", by W. J. Cody and W. Waits; published by Prentice Hall. The library currently includes:

sqrt() sin() cos() tan() asin()
acos() atan() log() exp()

The manual is nice. It details some of the attributes of the IEEE format, such as error conditions, and how this package (and Introl's runtime code) handle them. It has tables explaining exceptions, error reports and warnings reports. The repo ts signal the error or warning and the PC eddress that called the function and created the error.

Accuracy seems to be of prime importance to the author. Three pages of the manual are devoted to explaining how the package was tested, and the error limits found by this testing. There is also a table on "typical" execution times based on argument ranges. According to this table, which was generated on a CoCo (clock speed of 0.89Mhz), most of the functions execute in under 50ms.

The manual finishes with a method for locating the source of errors and warnings and a couple of test programs, one of which demonstrates error handling.

I personally look forward to using it. It should do a lot to help you users who really need scientific functions, but don't the expertise or time to develop your own package. I hope that it will eventually be available for the Windrush complier too. Incidently, in talking with the author after receiving the package, the subject of sources came up. They might also be made available at extra cost.

THE C PUZZLE BOOK

Our technical library at work got in a copy of "The C Puzzle Book" by Alan R. Feuer. The book was written to as a supplemental teaching aid to "The C Programming Language" by Kernighan and Ritchie. It is broken into six "chapters" covering different parts of the language.

If you are at all serious about the language consider this book a must. The puzzles are very short programs. It is the reader job to determine what the program will output to the terminal. They bring out both the power and quirks of the language and also give good illustrations of how, and how NOT, to program in C.

After you've tried a few you'll probably find yourself describing with adjectives like "simple", "clever" and "subtle." Indeed, I often described themas devious. What makes them devious is that despite their apparent simplicity, they require the user to be aware of the whole language. I think the author is purposely exploiting that chip that sort of grows on our shoulder when we see something apparently simple and say to ourselves "no sweat." Watch out, it isn't!

All the examples have been coded and run on the Unix C compilers for both the PDP-11 and VAX (1) computers. Even the ones that I just KNEW wouldn't. I won't give you my "score". Let's just say that I was chastened on more than one or two occasion. Most of the time I knew better, but didn't take the time to really think about them.

I am sure that you will find this book a challenge no matter how long you've been programming in the language. The author maintains that learning any programming language can be modeled in three steps.

Step one involves learning the syntax of the language to the point where a complier no longer complains of meaningless constructs.

Step two is to know how a complier will interpret properly formed constructions in the language.

Step three is to form a programming style that fits the language and results in clear, concise and correct programs.

In the words of the author: "The puzzles in this book are designed to help the reader through the second step. They will chanllenge the reader's mastery of the basic rules of C and lead the reader in seldom reached corners, beyond reasonable limits, and past a few open pits".

If you want to get the book, the complete information is

"The C Puzzle Book"
Alan R. Feuer
Prentice-Hall Inc.
1982

DISPATCHING

This month's code accomplishes nothing useful other than serving as one example of how to write a dispatch table in C. It is probably one of the more advance uses of the language that has been shown in this column to date in terms of language features.

We begin by defining a dispatch table. In this application, it is an array of structures. Each structure has three fields, an pointer to a command name string, a pointer to a command function and a pointer to a help functions that describes that command.

The program uses the table to look up a function that the user has requested by comparing the user's intput string to the functions named in the table. If a match occurs, then the program calls the command function with the rest of the input string as an argument.

You might ask why we would want to do it this way when the same thing could be accomplished with a series of "if-else if" statements. Well, there is one very good reason. Suppose that some time after a program of this sort is working you want to add a new function. And let's further suppose that this program has a built in command lister and help facility.

If the program has been built with "if-else if" constructs, then you must update the code in three different sections of the program; the command interpreter, the command lister and the help facility.

But if your program is using a dispatch table, then you have only to code the new functions and then add a structure new entry into the dispatch table. Since the table can be kept in a separate file by itself, you would only have to compile the new functions and the dispatch table; and then relink the program.

This example program has a command lister function called list() and a help facility called help(). Note that they both make use the dispatch table; nothing additional is needed. Let's look at the code a little more deeply.

The table, as was mentioned before, is an array in which each entry is a structure. So the first thing we need to do is to declare the structure. The declaration is

```
typedef struct dtable
        {
        char *tname,
        int (*cfunc)();
        int (*hfunc)();
        } DTABLE;
```

As a convenience for later in the program, the declaration was also typedef'd. This allows us to refer to the data type "struct dtable" with that name or with the symbol DTABLE. This is not necessary, but a nice convenience. The most interesting parts of the declaration are the two pointers to functions

```
int (*cfunc)();
int (*hfunc)();
```

These really say that cfunc and hfunc are pointers to functions that return int's. If you leave off the first set of parenthises you end up with

```
int *cfunc();
int *hfunc();
```

which would say that cfunc and hfunc returned pointers to int's. This is written up on page 116 of K&R (2)

Next we need to create an array of these structures and initialize it with the necessary pointers. This is done with

```
DTABLE distab[] =   {
                    "f1", f1, hf1,
                    "f2", f2, hf2,
                    "f3", f3, hf3,
                    "help", help, NULL,
                    "list", list, hlist,
                    "exit", exit, NULL
                    };
```

This declaration brings out a number of things. Note that number of elements in distab[] is not declared withing the brackets. The fact that we are initializing it makes the compiler happy since it will know the number of elements when the initialization is finished. Also note that we used DTABLE instead of "struct dtable". This was done for convenience. While it didn't buy us much here, in a larger program where a structure may be referenced many times, it can add clarity to the program and save typing.

To properly initialize the array, each structure, and every field of the structure, must have a value declared. Since there was no help function for help() or exit(), I used NULL. I can then make my code ignore NULL and not dispatch to it. Page 142 of K&R points out an alternative form which in this case would be

```
DTABLE distab[] =   {
                    {"f1", f1, hf1},
                    {"f2", f2, hf2},
                    {"f3", f3, hf3},
                    {"help", help, NULL},
                    {"list", list, hlist},
                    {"exit", exit, NULL}
                    };
```

Now we have a problem. What will the compiler think that the values f1 and hf1 etc really are? If it is being very strict, it may flag an error unless you predeclare them as function that return int's. This is done just prior to initializing the table. Again, you might not have to do it with some compilers. I have had to do it with some compilers but not others. I personally believe that you should though, for reasons of portability.

There are two ways to traverse through the array. We can either know how many entries are in the table and exit execute a loop that many times, or add a "sentinal" value for the last entry and iterate the loop until the sentinal value is found. I chose to use the fo mer since the sizeof() compiler directive makes it so easy. We only need to declare an initialized integer that contains the number of entries. This is done with

```
int table_size = sizeof(distab) / sizeof(DTABLE);
```

The sizeof() operator tells the compiler to compute the initialized size of distab[] and divide it by the computed size of the structure. This gives the number of elements in the array. Nifty.

Now we can write a simple functions to access the table. This is done in three places in the program. The command interpreter looks for a match of the user's command with the command names. The command lister just dumps out all the command names in the table. The help dispatcher gets the next token and looks it up in the list. If there is a match and the pointer to the function's help discriptor, hfunc, is not set to NULL then it calls the descriptor function.

I think that you will find the program straight forward and easy to understand. Like I said earlier, this program does nothing useful, but serves as a testbed for a lot of different and more advanced C features. It has been compiled and tested on the Windrush compiler.

IT'S A WRAP

That's it for this month. The next two columns will deal with a modem program written in C. After all, you have them in assembler, Basic and Pascal so why not C. The next colmn will be a simple version that allows you to use the terminal transparently and to capture the incoming data to a buffer, which can then be saved. After second colmn will have Ward Christianson's XMODEM protocol, a nifty packet system for transfering files between machines. Till then.

NOTES

(1) PDP-11 and VAX are trademarks of the Digital Equipment Corporation.

(2) "K&R" refers to "The C Programming Languege" by Kernighan and Ritchie, published by Prentice-Hall.

```
/*
 * dispatch.c
 * n n comno
 *
 * A test bed for testing dispatch tables
 * driven command interpreters
 */

#include <stdio.h>
#include <ctype.h>


#define MATCH 0
#define LINE 80
#define TOK 20
#define FOREVER for(;;)


/*
 * this is the dispatch structure
 */
typedef struct dtable
        {
        char *fname;
        int (*cfunc)();
        int (*hfunc)();
        } DTABLE;

/*
 * here are the definitions of the functions
 * that will be pointed to
 *
 * the need for this section may be compiler
 * dependant
 */
int f1(), hf1();
int f2(), hf2();
int f3(), hf3();
int help();
int list(), hlist();
int exit();



/*
 *  nd here is the dispatch table
 * initialization
 */
DTABLE distab[] = {
                "f1", f1, hf1,
                "f2", f2, hf2,
                "f3", f3, hf3,
                "help", help, NULL,
                "list", list, hlist,
                "exit", exit, NULL
                };


int table_size = sizeof(distab) / sizeof(DTABLE);
main()
    {
    char buff[LINE], token[TOK], *next, *getoken();
    int i;

    /*
     * just sit here and do whatever
     * the user requests
     */
    FOREVER
        {
        printf(">>> ");
        gets(buff);
        next = getoken(token,buff);
        for (i = 0; i < table_size; i++)
            if (strcmp(token,distab[i].fname) == MATCH)
                {
                (*distab[i].cfunc)(next);
                break;
                }

        if (i == table_size)
            printf("Illegal command\07\n");
        }
    }



/*
 * this function will list all the available
 * commands, it just dumps out the command
 * names that are in the table
 */
list()
    {
    int i;
    skip(2);
    printf("Available commands are:\n\n");
    for (i = 0; i < table_size; i++)
        printf("    - %s\n",distab[i].fname);
    skip(2);
    }

hlist()
    {
    printf("LIST -- list out all the available commands\n\n");
    }

/*
 * the help function will list dispatch to
 * a help discriptor if one exits, otherwise
 * it will print out the commands for which
 * there is help
 */
help(s)
    char *s;
    {
    char token[TOK];
    int i;

    getoken(token,s);
    skip(2);
    for (i = 0; i < table_size; i++)
        if (strcmp(token,distab[i].fname) == MATCH)
            if (distab[i].hfunc != NULL)
                {
                (*distab[i].hfunc)();
                skip(2);
                return;
                }
            else
                break;
    printf("Help is available for:\n\n");
    for (i = 0; i < table_size; i++)
        if (distab[i].hfunc != NULL)
            printf("    - %s\n",distab[i].fname);
    skip(2);
    }



/*
 * output blank lines to
 * the terminal
 */
skip(n)
    int n;
    {
    while (n--)
        putchar('\n');
    }

/*
 * Gets then next token available in "from"
 * and puts it into "to" with no leading or
 * trailing white space.
 *
 * RETURNS      pointer to next character in
 *              from if a token was found
 *
 *              NULL if no token was found, and
 *              will also set to[0] to NULL
 */
char *getoken(to,from)
        char *to,       /* token buffer */
             *from;     /* source buffer */
        {

        /*
         * zero the token just in case
         * and test for NULL line
         */
        *to = '\0';
        if (from == NULL  ||  *from == '\0')
                return(NULL);

        /* skip leading white space */
        while (*from && isspace(*from))
                (from++);

        /* tranfer the token, converting to lower case */
        while (*from  &&  !isspace(*from))
                (*to++ = tolower(*from++));

        *to = '\0';
        return(from);
        }


/*
```

```
 * here are some test functions that
 * serve only to make the dispatch
 * table a little bigger
 */
f1(s)
    char *s;
    {
    printf("f1() has the arguments \"%s\"\n",s);
    }

hf1(s)
    char s;
    {
    printf("f1() serves no useful purpose\n");
    }
f2(s)
    char *s;
    {
    printf("f2() has the arguments \"%s\"\n",s);
    }

hf2()
    {
    printf("f2() serves no useful purpose\n");
    }


f3(s)
    char *s;
    {
    printf("f3() has the arguments \"%s\"\n",s);
    }

hf3()
    {
    printf("f3() serves no useful purpose\n");
    }
```

DATA STRUCTURES

Data Structures:
An Introduction to Data Base
and Record Management Systems

by E. M. (Bud) Pass, Ph.D.
Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers GA 30207
Telephone Number 404-483-1717/4570

INTRODUCTION

The use of data base and record management systems is becoming more important, and yet many do not understand what they are, how they work, and what they can and cannot do. This article presents an introduction to data base and record management systems. It provides an example of the construction of a record management system and also discusses the differences between data base and record management systems.

DATA BASE AND RECORD MANAGEMENT SYSTEMS

The term "data base" is often applied to cases in which it should not be used, adding to the confusion of the term. A data base is a body of information and a "data base system" is a set of mechanisms defining operations on that body of information. However, not every set of operations on a body of data may be considered as a data base system.

Record management systems provide the mechanisms for the storage and retrieval of the records containing information, but with little regard for the contents of the records, as opposed to data base systems, which are designed with greater regard for the contents of the file or files comprising their data storage.

Many record management systems are incorrectly termed "data base systems". Record management systems range from trivial Rolodex and 3 by 5 cards, thru more complicated mailing lists, accounting systems, and inventory systems, to complex VSAM based storage and retrieval systems on mainframes.

Data base systems are often based upon record management systems, although they are not necessarily based upon them, and they are far more complex. They provide complex mechanisms for the flexible storage and retrieval of information in manners not specifically envisioned by the designers of the data base system. A data base system is capable of relating the bits of information it stores to one another, and is affected by the meaning of that information.

According to this definition, most so-called "data base" systems are not really data base systems, but are record management systems. This includes dBase II, Infomag, UDRI, XDMS, RMS, and many others advertised as "data base" systems.

This is not to trivialize record management systems. Some are quite complex in their capabilities, and most provide flexible and useful functions to their users. Since they are important, they will be defined and discussed.

RECORD MANAGEMENT SYSTEMS

A record management system maintains a body of records. These records contain information describing something of interest to the user. The file or files managed by the system may each be organized by one or more keys, which are composed of data contained in the records.

Organizing a file alphabetically is an implicit recognition of a traditional ordering of a file by a name or other significant field, such as a part number. Files may be organized by a key representing one or more fields, such as name or zip-code and name, or they may be organized by more than one key, each of which represents one or more fields in the record.

A file organized by one key may or not posess an index, but a file organized by more than one key must posess an index. An index is a "table of contents" of a file, providing a complete or partial list of the keys by which a file is organized, and a record pointer by which a given record may be located. A file organized by only one key does not necessarily need an index, since it may be searched randomly with a binary search algorithm, or it may be alternately scanned sequentially for the desired record or records.

While the phrases "data base system" and "record management system" sound technical and sophisticated, they are actually based upon some very simple concepts.

As alluded to earlier in this article, a Rolodex card file actually forms a part of a very simple form of record management system. The individual cards represent data records, and they are normally arranged alphabetically by company name or last name. No index is required, and the human user provides all searching, updating, deleting, adding, and other manipulation of the records, in addition to the interpretation of the data written on the cards.

Some computerized record management systems are no more sophisticated than the Rolodex card system, providing only very rudimentary operations on the records under their management. Such systems are characterized by sequential filing systems supporting only one key, and by limited update, selection, and formatting capabilities.

EXAMPLE RECORD-ORIENTED FILE CREATION

Suppose it is necessary to create a mailing list system mirroring a Rolodex card file. It is clear that the records in a simple mailing list would correspond on a many-to-one basis to the original Rolodex cards. Consider what is normally written on a typical card, as

follows:

Generic Electronics Company
10 Main Street
Nowheresville, MI 12345-6789
(123) 555-1212 ext 234
John S. Generic, President
I. M. Slippery, Salesman
Murphy Slow, Technician

The meaning of each line on the Rolodex card is usually clear to the human processing the file. If the data on the card were entered directly into a record management system, chaos would result, since the interpretations of the fields would be ambiguous, as the record processing system has no way to distinguish one field from another.

Someone must indicate to the record management system how to place selected data into retrievable fields in the records on the file. A formatting operation, almost always accomplished by a human entering data from the card into a structured form or screen, is usually required to structure previously unstructured data, as in the case of the Rolodex card file.

How does one structure a record? In a record management system, the level of detail required in the fields of a record is specified by the envisioned usage of the data in the records. If all that is required of the data on the Rolodex cards is a telephone list, then all that would probably be entered and stored is the names and telephone numbers. If the telephone numbers are irrelevant, as in the case of a mailing list for a solicitation, they would probably not be entered or stored.

In many cases, however, all the information on the source document is entered and stored, even if not all of the data is of immediate interest. The theory is that the data is easier to capture all at once than it is to capture portions of it, multiple times.

Those fields of data to be entered on the screen or form must be defined. The obvious manner of entering data, a line at a time, is no better than entering it in an unformatted manner. The name fields must be separated from the address fields, which must be separated from the city, state, and zip-code fields, which must also be separated from the telephone number and information fields, etc. Then, the record management system can store the fields in their appropriate slots.

RECORD MANAGEMENT OPERATIONS

Once the data has been successfully captured, it is useless unless it can be used. The record management system must provide at least a small number of rudimentary capabilities to enable the manipulation of the records.

Managing a set of records involves the following primitive operations:

- add new records
- edit contents of records
- delete records
- search records on selected fields
- format selected fields to printer or other files

Managing a set of records may also involve non-primitive operations such as the following:

- maintenance of multiple key access
- logical operations on selected fields
- perform computations on selected fields
- summarize selected fields
- sort selected records on specified keys
- generate reports with field breaks

Thus, record management systems may provide significant capabilities, often sufficient for many capabilities, without being truly classified as data base systems.

DATA BASE SYSTEMS

A data base system may be designed and used on one or more of the following levels:

- internal
- external
- virtual

The internal level of data base system representation is concerned with the details of the storage media used, and of the organization of the record management or other system which manipulates the data itself on the storage media. It is also concerned with the terminal management and other systems which allow the user to communicate with the system. The users of a data base are almost never concerned with this level, but the original designers and the maintainers of the mechanisms supporting the data base system are very concerned with it. Much of the efficiency of a data base system is based upon the internal level, since encoding, storage, and retrieval are the major activities at this level.

The external level of data base system representation provides a view of the information in a data base, as oriented toward a particular use of the data base system. This view may be derived from the results of an application program used to store and retrieve data into and from the data base, or it may be derived from the results of inquiries made of the data base at a particular time. The users and system analysts of a data base system are generally concerned with this level, and with the next one.

The virtual level of data base system representation connects the information stored in the data base managed by the data base system to the real-world system. The real-world system modelled by a corporate financial data base would be quite different from the real-world system modelled by an airline scheduling system. The users, system analysts, and user management are concerned with this level of data base system representation.

As noted earlier, many data base systems are based upon record management systems. In such a data base system, extensive use is made of multi-keyed records, to the point of being able to store data and to retrieve it by any logical combination of keys.

A requirement on the operations of data base systems which separates them from the simpler record management systems, is the necessity for the simple addition of new types of data, or fields, and the associated keys and indices, without the complete redefinition and regeneration of the data base, as is required by many record management systems. This capability is called "extensibility". Some of the more complex record management systems have this ability.

Data base systems usually support an extensive selection language, allowing the specification of Boolean qualifications for selected records, and the further processing of the selected and rejected records. They also usually support successive grouped data base selections, in which the results of one selection process may be saved and reselected by new, sharper, or different criteria.

A data base selection language often has statements such as the following:

```
(state is 'MI' or 'AK') and
(name is 'GENERIC?' or '?ELECTRONICS?')
```

in which the terms "state" and "name" must be associated with the fields in the data base thru a dictionary. The question marks in the query literals represent the matching of any string of characters in the corresponding fields in the records being searched in the file.

More than one file may be involved in a search operation, as a data base may easily be composed of many cross-related files. For example, consider a financial data base containing all of the general ledger, accounts receivable, accounts payable, payroll, and inventory files of a corporation. In addition to linking the five files, the data base system may be required to be capable of relating current and historical data from the files and to produce projections under a model programmed in a special inquiry language.

Data base systems provide extensive protection for

the data bases under their management. This includes inquiry and update security, simultaneous and/or sequential multi-user access, automatic update journalling and update backout, etc. Almost none of these abilities are present in record management systems.

For very complex data base requirements, non-record-oriented data bases, sometimes called relational data bases, are used. Examples of the use of such data bases would be for weather prediction, military, medical, airline ticket scheduling, space research, and for other cases in which the structure of the data base cannot be predicted in advance, or in which the operations on the data base are not to be restricted to a limited number of functions.

The combination of a relational data base and an advanced query language, such as a subset of A A, implements a form of a so-called "expert" system. In these systems, the user formulates questions of the system, or makes long-standing requests of the system to be notified upon the occurrence of some condition or combination of conditions. In return, the system may determine that the information under its control is insufficient to satisfy the request and ask the user for other information or to restate the request.

As could be deduced, these systems are generally all still very new and still under research and development, with limited use. Full implementations require the largest mainframes and clusters of mainframes to implement the requests in a reasonable amount of time.

At least one true relational data base system has been implemented on a micro, and several systems being called "expert" systems have recently appeared. These are generally only partial implementations and their capabilities are highly restricted. However, the need is present for such systems and more advanced ones will certainly appear.

SUMMARY

This article has presented an introduction to the concepts of record management systems and data base systems. It noted that essentially all of the existing micro-based systems claiming to be data base systems are really record management systems, and are not data base management systems.

USING THE MM58167 REAL TIME CLOCK

James H. Gross Jr.
4930 New Castle Rd.
Lowellville,Oh. 44436
Phone 216-536-6760
Janurary 25,1984

Editor
'68' Micro Journal
5900 Cassandra Smith Rd.
Hixson, Tn. 37345

Dear Mr. Williams,

I am sending you on the enclosed diskette an article that I would like to have considered for publishing in '68' Micro Journal. Also on that diskette are copies of this letter and text files for the programs listed in the article. All files are compatable with either the TSC editor or Stylograph.

'68' Micro Journal has been invaluable in helping to get the most from my 6809 system. I have selected copies of '68' Micro Journal going back to 1979. The recent RELINK utility was one of the most useful programs that you have published. I just hope that the article I have submitted to you will help or entertain others as much.

The system that I am using is based on the FEBE group mother board and viedo card. The cpu is the Smoke Signal SCB-69. Disk operation is controlled by a Smoke Signal DCB-4A. RAM cards are by Smoke Signal and Digital Research. Disk drives are Tandon a 100-2 and an 848-2. The printer is an MPI 88T.

I have been involved in building and using microprocessor systems since 1976, things sure have changed since the original Altair 88 0 kit came out. I built my first 680 system in 1978, of course it was SWTP. I didn't switch to the 6809 until last year. Just to place things into prespective, the peolpe who purchase the Apple computer are buying a 6502 a processor that is as old as the 6800. The point is that we really haven't explored the power of the chips that we have available.

Well, in closing I would like to reiterate how much I enjoy '68' Micro Journal. If you like this article I will be very glad to share the results of some other projects that I am working on.

Yours sincerely,

James Gross

Ed's Note: James, thanks for this article and looking forward to artic es about some of your other projects.

DMW

- - -

USING THE MM58167 REAL TIME CLOCK

by James H. Gross Jr.
4930 New Castle Rd.
Lowellville, Ohio 44436

the smoke signal scb-69 has an on board rtc. however, when running flex there isn't any application software to use it. a little time spent studying the data sheet reveals that the MM58167 is easy to use and applications programs can be written in high level languages such as lecidata's Pascal version 3.

the MM58167

The National Semiconductor MM58167 is a CMOS device that will function as a real time clock calendar when properly interfaced to a microprocessor system. The device includes counters, latches, interrpts, status, and other system registers. The data presented to and from the MM58167 is in BCD(binary coded decimal) format. The status register is used to detect counter rollover durning a read operation and should be tested after each read. The MM58167 has a special register named 'GO', this register is used to reset and start the seconds, tenths, hundereds and thousandths counters so that the RTC can be started at a known time.

the SCB-69

Smoke Signal's SCB-69 is one of the finest 2 MHz. 6809 processor boards. The board has has many options, and features but what we are interested in is that it provides a ready made hook for the MM58167. To operate the RTC several switches must be configured, s2-7 on, s2-8 on, and the board may be running at either 1 or 2 MHZ.(W1-2,3). The SCB-69 locates the MM58167 registers starting at $F700.

display real time clock

When looking at the MM58167 the thing to do first is see if you can access the registers on the chip. So, the first program that should be tried is one to display the registers in MM:HH:SS:TT format.
The program called 'clock' is a Pascal program and was configured to run as a FLEX command.This program will demonstrate the basic method of addressing the registers and the method used to read a register.

```
P-6800 RUN-TIME SYSTEM  V 3.9:7   : COPYRIGHT C 1982 LUCIDATA
USABLE CONTIGUOUS MEMORY  $C000
CURRENT STACK RESERVATION $2200

PASCAL P-COMPILER : VERSION 3.M   : COPYRIGHT C 1983 D.R.GIBBY

   0 PROGRAM CLOCK ;
   0
   0 (A program to use the National Semiconductor MM58167 RTC. Which
   0   is part of SSB's SCB-69. To display the time in hh:mm:tt
   0   format.)
```

```
  0
  0 VAR
  0 (*$A=$F700*)(location of 58167 registers)
  0 MILSEC,CENTISEC,SEC,MIN,HRS : BYTE ;
  0
  0 (*$A=$F714*)(location of 58167 'status' byte)
  0 STATUS : BYTE ;
  0 (*$S*)(return to stack allocation of variables)
  0
  0 TSEC,H,S,T,M : BYTE ;
  0
  0 FUNCTION CONVERT(X:BYTE):BYTE ;
  4 (Convert BCD to standard binary numbers)
  4 BEGIN (CONVERT)
  4   IF X>09 THEN X := X-6 ;
 40   IF X>20 THEN X := X-6 ;
 76   IF X>30 THEN X := X-6 ;
112   IF X>40 THEN X := X-6 ;
148   IF X>50 THEN X := X-6 ;
184   IF X>60 THEN X := X-6 ;
220   IF X>70 THEN X := X-6 ;
256   IF X>80 THEN X := X-6 ;
292   IF X>90 THEN X := X-6 ;
328   CONVERT := X ;
344 END ; (CONVERT)
348
348 PROCEDURE warms ; EXTERNAL $CD03 ; (FLEX warm start entry
348                                     point.)
348 BEGIN (CLOCK main)
352
352 WRITE(CHR(30),CHR(2)); (turn off cursor)
376
376 REPEAT
376 TSEC := SEC ; (read sec counter)
396 UNTIL STATUS <> 1 ; (test for counter roll over)
416
416 TSEC := CONVERT(TSEC) + 35 ; (convert bcd to standard binary
452         numbers and set display for 35 seconds)
452
452 IF TSEC > 59 THEN TSEC := TSEC - 60 ; (adjust for minute
488                                         crossover)
488 REPEAT (main display loop)
488
488   REPEAT
488     H := HRS ; (read hours counter)
508   UNTIL STATUS <> 1 ; (test for hours roll over)
528   H := CONVERT(H) ; (convert bcd to standard binary numbers)
560
560   REPEAT
560     M := MIN ; (read minute counter)
580   UNTIL STATUS <> 1 ; (test for minute roll over)
600   M := CONVERT(M) ; (convert bcd to standard binary numbers)
632
632   REPEAT
632     S := SEC ; (read seconds counter)
652   UNTIL STATUS <> 1 ; (test for seconds roll over)
672   S := CONVERT(S) ; (convert bcd to standard binary numbers)
704
704   REPEAT
704     T := CENTISEC ; (read tenths/hunderedths counter)
724   UNTIL STATUS <> 1 ; (test for tenths/hunderedths roll over)
744   T := CONVERT(T) ; (convert bcd to standard binary numbers)
776
776   WRITE(H:2,':',M:2,':',S:2,':',T:2,CHR(13)) ; (display time
840                                                 registers)
860 UNTIL TSEC = S ; (display duration test)
880
880 WRITELN(CHR(30),CHR(5)); (restore cursor)
908
908 warms ; (Jump to warm start entry point.)
916
916 END . (CLOCK main)
920 BYTES
END OF PASS 1
END OF PASS 2
OK  TO RUN

END OF PROGRAM EXECUTION.
```

The comments in the program will explain how it works. The 'status' register is set to binary 1 if a read occurs during a counter rollover. The program tests for this condition by using REPEAT-UNTIL loops forcing the program to reread a register that has rolled over. The program uses the FLEX warm start entry point to terminate program execution, this prevents the END OF PROGRAM message from being displayed.

set the Real Time Clock

The program 'clock' will read and display the MM58167 counter registers, but that is of little use if the incorrect time is displayed. The program called 'clockset' will allow any legal 24 hr. time to be entered and be used as the starting point for the RTC.

```
P-6800 RUN-TIME SYSTEM  V 3.9:7   : COPYRIGHT C 1982 LUCIDATA
USABLE CONTIGUOUS MEMORY  $C000
CURRENT STACK RESERVATION $2200

PASCAL P-COMPILER : VERSION 3.M   : COPYRIGHT C 1983 D.R.GIBBY

   0 PROGRAM clockset ;
   0
   0 (The purpose of this program is to set the counters of the
   0  MM58167 RTC which is used on the SSB's SCB-69 CPU card. By the
   0  use of the 'go' command register the RTC can be set to start
   0  at an exactly known time.)
   0
   0 VAR
   0
   0 loGA=$F703*)(Address variables to MM58167 registers.)
   0 min,hrs,day,date,mon : BYTE ;
   0
   0 (*$A=$F712*)
   0 creset,lreset,status : BYTE ;
   0 go : CHAR ;
   0 (*$S*) (Variable allocation returned to the stack.)
   0
   0 month,dayofmon,dayofweek,hours,minutes : BYTE ;
   0
   0 FUNCTION translate(x:BYTE):BYTE ;
   4 (Convert standard binary numbers to BCD.)
   4 BEGIN
   4   IF x>90 THEN x := x+6 ;
  40   IF x>80 THEN x := x+6 ;
  76   IF x>70 THEN x := x+6 ;
 112   IF x>60 THEN x := x+6 ;
 148   IF x>50 THEN x := x+6 ;
 184   IF x>40 THEN x := x+6 ;
 220   IF x>30 THEN x := x+6 ;
 256   IF x>20 THEN x := x+6 ;
 292   IF x>09 THEN x := x+6 ;
 328   translate := x ;
 344 END ;
 348
 348 PROCEDURE warms ; EXTERNAL $CD03 ;(FLEX warm start entry point)
 348
 348 BEGIN (clockset main)
 348
 352 (Get time data for later loading into RTC registers.)
 352 WRITELN ;
 356 WRITE('Enter month, 1-12 ') ;
 380 READLN(month) ;
 396 WRITELN ;
 400 WRITE('Enter day of month, 1-31 ') ;
 432 READLN(dayofmonth) ;
 448 WRITELN ;
 452 WRITE('Enter day of week, Mon=1 to 7 ') ;
 488 READLN(dayofweek) ;
 504 WRITELN ;
 508 WRITE('Enter hours, 1-24 ') ;
 532 READLN(hours) ;
 548 WRITELN ;
 562 WRITE('Enter minutes, 0-59 ') ;
 580 READLN(minutes) ;
 596 WRITELN ;
 596
 600 (Convert time data to BCD and load into RTC registers.)
 600 mon := translate(month) ;
 636 date := translate(dayofmonth) ;
```

```
672 day := translate(dayofweek) ;
708 hrs := translate(hours) ;
744 min := translate(minutes) ;
744
780 {Set up to use 'go' register}
780 WRITE('Enter a character to start clock ') ;
820 READ(go) ; {Place the 'go' register address on the address bus}
820           {The clock is started}
820
832 warms ; {Jump to FLEX warm start}
832
840 END . {clockset main}
844 BYTES
END OF PASS 1
END OF PASS 2
OK  TO RUN

END OF PROGRAM EXECUTION.
```

Just to prove that the correct time(or any time that was entered) is really there run the 'clock' program.

replacing the FLEX date prompt

Now that a method has been developed to access the MM58167 registers, we should try to do something useful. The purpose of the program 'putdate' is to use the MM58167 to place the proper date in the FLEX date registers when the system is booted, the program then executes the startup file and jumps to the warm start entry point. Just a word of caution, if you are not familar with the FLEX system read the ADVANCED PROGRAMMERS MANUAL for FLEX before you attempt to modify or to overlay parts of FLEX.

```
P-6800 RUN-TIME SYSTEM  V 3.9:7   : COPYRIGHT C 1982 LUCIDATA
USABLE CONTIGUOUS MEMORY  $C000
CURRENT STACK RESERVATION $2200

PASCAL P-COMPILER : VERSION 3.M   : COPYRIGHT C 1983 D.N.GIBBY

  0 PROGRAM putdate ;
  0 {The purpose of this program is to provide the flex system with
  0 the date data from the National Semiconductor's MM58167 RTC
  0 which is part of the SSB's SCB-69. A three byte assembly
  0 language program is then overlayed on the flex system to disable
  0 the date prompt and jump to this program.}
  0
  0 TYPE
  0   STRLEN = ARRAY[1..16]OF CHAR ;{length of the command}
  0
  0 VAR
  0
  0 {=$A=$C080} {Start of flex system's command line buffer}
  0 flexcmd : STRLEN ;
  0 {=$A=$C090} {Address of the last byte of the flex command we
  0                                                   will use}
  0 flexnul : BYTE ;
  0 {=$A=$CC14} {address of the flex command line buffer pointer}
  0 flexptr : INTEGER ;
  0 {=0A=$CC0E} {Start of the flex system date registers}
  0 flexmonth : BYTE ;
  0 flexday   : BYTE ;
  0 flexyear  : BYTE ;
  0
  0 {=$A=$f704} {Address of the needed MM58167 registers on the
  0                                                   SCB-69}
  0 day,month : BYTE ;
  0 {=$5e} {return to stack variable allocation}
  0
  0 FUNCTION CONVERT(x:BYTE):BYTE ;
  4 {Convert bcd to standard binary numbers}
  4 BEGIN
  4   IF x>09 THEN x := x-6 ;
 40   IF x>29 THEN x := x-6 ;
 76   IF x>39 THEN x := x-6 ;
112   IF x>49 THEN x := x-6 ;
148   IF x>59 THEN x := x-6 ;
184   IF x>69 THEN x := x-6 ;
220   CONVERT := x ;
236 END ;
240
```

```
240 PROCEDURE DOCMD ; EXTERNAL $C04B ;{FLEX routine to execute a
240                    command already in the FLEX command buffer}
240
240 PROCEDURE WARMS ; EXTERNAL $C003 ;{FLEX warms entry point}
240
240 BEGIN {putdate main}
244 flexmonth := CONVERT(month) ;
284 flexday := CONVERT(day) ;
324 flexyear := 84 ;
340 flexcmd := 'EXEC,STARTUP.TXT' ; {Flex commands we wish to
340                                                  execute}
368 flexptr := $8080 ; {set line buffer pointer to first character
368                                                   in buffer}
380 flexnul := 13 ;{last character in command line}
396 DOCMD ;{execute command in command buffer}
404 WARMS ; {jump to FLEX warm start entry point}
412 END . {putdate main}
416 BYTES
END OF PASS 1
END OF PASS 2
OK  TO RUN

END OF PROGRAM EXECUTION.
```

The following steps will produce a copy of FLEX.SYS that will boot and also place the date in the date registers.

Step one: Work only with a copy of your system disk.
Step two: Rename FLEX.SYS, FLEX9.BIN.
Step three: Copy the FLEX9.BIN file to drive 1 from drive 0.
Step four: Compile the PUTDATE.TXT file using Lucidata Pascal ver. 3.
Step five: Using Lucidata Runtime system execute the the following command line: RUN3,PASCAL3,PUTDAT,$C this will provide the following information: start address, stop address, and transfer address of the combined RUNTIME system and putdate program. Remember the transfer address.
Step six: Save the file from start to stop address, do not use a transfer address. The file should be on drive 1 and have a name 1.PUTDATE.OVL.
NOTE: The following files should be on drive 1:
FLEX9.BIN
PUTDATE.OVL
Step seven: Assemble the following 6809 mnemonics.

```
ORG $CA02 flex date prompt entry point
JSR $11EB transfer address from step 5
END
```

Name the file DATEOVL.BIN and place it on drive 1.
At this point the three files on drive 1 are:
FLEX9.BIN
PUTDATE.OVL
DATEOVL.BIN
Step eight: Execute the following Flex command line:

```
APPEND,1.PUTDATE.OVL,1.FLEX9.BIN,1.DATEOVL.BIN,0.FLEX.SYS
```

This step will have placed the new FLEX.SYS file on drive 0.
Step nine: Link the new FLEX.SYS file on drive 0.
Step ten: Place the FLEX command 'DATE' in the STARTUP.TXT file and FLEX will print the date that was put in the date registers.

If all has gone well you have a version of FLEX that loads a Pascal program executes that program and returns to FLEX when a system boot is executed. Just another word of warning, not all FLEX systems are standard so check your documentation to get the proper entry point for the date prompt.

time that sieve

Ever since Gilbreath had his Eratosthenes sieve program and bench marks published in the January 83 BYTE magazine, that program has become the defacto standard against which the speed of a systems processing is measured. There have been many questions raised and many disagreements over some of the timings in that article. The 'primesieve' program uses the stopwatch function of the MM58167 to time the execution of the benchmark program. After the sieve portion of the program is run the exectution time in seconds is printed.

```
P-6800 RUN-TIME SYSTEM  V 3.9:7   : COPYRIGHT C 1982 LUCIDATA
```

```
USABLE CONTIGUOUS MEMORY  $C000
CURRENT STACK RESERVATION $2200

PASCAL P-COMPILER : VERSION 3.M   : COPYRIGHT C 1983 O.A.GIBBY

   0 PROGRAM PrimeSieve ;
   0
   0 {The purpose of this program is to benchmark the execution of
   0  the Eratosthenes Sieve Prime Number program. This program used
   0  the RTC on board the Smoke Signal SCB-69 to function as a stop
   0  watch. Eratosthenes Sieve program by Gilbreath, Byte
   0  Publications, January 1983.)
   0
   0 CONST
   0 size = 8190 ;
   0
   0 VAR
   0
   0 {Erathosthenes sieve variables}
   0 flags : ARRAY [0..size] OF BOOLEAN ;
   0 i,prime,k,count,iter : INTEGER ;
   0
   0 {MM58167 RTC registers}
   0 {$A=$F700}
   0 millisec,centisec,sec,min : BYTE ;
   0 {$A=$F712}
   0 creset,lreset,status,go : BYTE ;
   0 {$S0}{return to stack allocation}
   0
   0 csec,csec,seconds,minutes : BYTE ;
   0
   0 FUNCTION convert(x:BYTE):BYTE ;
   4 {convert BCD to standard binary numbers}
   4 BEGIN {convert}
   4   IF x>99 THEN x := x-6 ;
  40   IF x>20 THEN x := x-6 ;
  76   IF x>30 THEN x := x-6 ;
 112   IF x>40 THEN x := x-6 ;
 148   IF x>50 THEN x := x-6 ;
 184   IF x>60 THEN x := x-6 ;
 220   IF x>70 THEN x := x-6 ;
 256   IF x>80 THEN x := x-6 ;
 292   IF x>90 THEN x := x-6 ;
 328   convert := x
 328 END ; {convert}
 348
 348 BEGIN {main program block}
 352 {start timer}
 352 creset := 8 ;
 368 go := 0 ;
 384   WRITELN('10 iterations') ;
 408   {Eratosthenes sieve program}
 408   FOR iter := 1 TO 10 DO BEGIN
 428     count := 0 ;
 436     FOR i := 0 TO size DO
 448         flags[i] := TRUE ;
 492     FOR i := 0 TO size DO
 504         IF flags[i] THEN BEGIN
 528             prime := i + i + 3 ;
 546             {WRITELN(prime) ;}
 548             k := i + prime ;
 564             WHILE k <= size DO BEGIN
 576                 flags[k] := FALSE ;
 592                 k := k + prime
 596             END ;
 612             count := count + 1
 620         END ;
 644   END ;
 664     WRITELN(count,' primes') ;
 692   {read timer}
 692   REPEAT
 692   minutes := min ;
 712   UNTIL status <> 1 ;
 732   REPEAT
 732   seconds := sec ;
 752   UNTIL status <> 1 ;
 772   REPEAT
 772   csec := centisec ;
 792   UNTIL status <> 1 ;
 812 {do bcd to standard binary conversion}
 812   minutes := convert(minutes) ;
 844   seconds := convert(seconds) ;
 876   csec := convert(csec) ;
 908   WRITELN('execution time: ') ;
 936   WRITE(60*minutes+seconds+(0.01*csec):6:2,' sec.') ;
1004 END .{main program block}
1008 BYTES
 END OF PASS 1
 END OF PASS 2
 OK  TO RUN

 END OF PROGRAM EXECUTION.
```

When this program was run using the SCB-69 it executed in 251.04 sec. This is very good performance for a P-code Pascal system. Lucidata's Pascal ver 3. is a very fine compiler. As demonstrated by these programs it is possible to do meaningful systems level programming using Lucidata's Pascal. Remember to reset the RTC using 'clockset', after running 'primesieve'.

conclusion

I have often been asked by friends just what good a RTC was. Well, as you can see there are many applications for a real time clock. It is not so much what has been with this device but what can be done if creative and inventive programmers look for uses. One application that is being looked at is to use the RTC to control a series of devices in real time, using the B side of the MP-1A in I/O port 6.(the printer port is FREE)

I will welcome any letters and comments that you may have and will answer as many as possible, It is not possible for me to provide listings on any media of the programs in this article.

UNIFLEX GENERAL LEDGER REVIEW

Brian Frank
Intrigue/Compubyte
283 Dufferin Road
Montreal, Canada
H3X 2Y4

November 1, 1983

Uniflex Software Product Review

SPECTRA SYSTEMS

General Ledger Package

This review is being written under duress. Having received the above-mentioned package in April of this year after its initial release, I had 'panned' it somewhat. The review was forwarded to SPECTRA for their comments, and a 'new' version of the software, complete with a commented list of the implemented changes accompanying.

In particular, I had made nasty comments about the system documentation which was provided. The authors had provided not one, but two individual copies of a somewhat sketchy manual with the initial release. No binder had been supplied.

The new version provided a similar documentation package. This package, while extremely 'user friendly', does not contain any accounting tips or "how-to's" of computer accounting. A result of this limitation leads me to suggest that anybody intending to purchase this package also be familiar with basic accounting techniques or purchase a textbook on the subject.

This package is written entirely in T.S.C. Cobol, with the required support utilities being provided. You will however require at least the T.S.C. Sort Merge Package in order to make this system functional. It is additionally recommended that one acquire the Enhanced Printer Spooler package, although this is not a pre-requisite.

Installation of the package will yield a number of new (or expansion of) two new directories. These are '/programs' and '/programs/gen-ledger'. A number of new programs will appear in the '/bin' and '/etc' directories, mostly involving Cobol support utilities.

Installation on one's system is as is customary with UNIFLEX systems. One mounts the supplied floppy disk as '/usr2' and then simply types '/usr2/install'. The rest is automatic.

The termcaps function is a utility that is supplied for all UNIX systems and has been emulated for use by the T.S.C. Cobol package. This function relieves the programmer from the worry of terminal configuration. It would be nice if all system I/O was passed through termcaps (selectively of course).

The next procedure (as outlined in the manual) is to format the termcaps file for your terminals as they appear on the various ports of your system. A

coeprehesslve dlssertatloo oo the how'e of why's of terminal configuration as well as sample terelosls is given. Among those supplied are the ct-82, the adm3 and so on. If your terminal is not supplied, simply edit in the screeo parameters for it.

One word of note is that the wait after screen clear delay may not be set to '0'. This parameter's parsing routine subtracts one from the supplied number to calculate the delay. An entry of '0' generates a 255 second delay after a screeo clear !! This particular action really should have been documented by T.S.C. or should have been added by SPECTRA.

Additionally, some of the progress require that a 24 line terminal be used. Thus, the original CT-82 from Southwest is not the terminal of choice.

Some features of the package contribute to making this the simplest g-l package I have ever had the pleasure of installing on any system. Among the features is an installation program which allows for the usage of default account ranges or user defined ranges for report generatioo. All reports are generated based upon this installatioo range specification. Thus, no clever (difficult) report spetification files need be prepared though complete specifications are provided for those who may wish to attach and re-organize the reports.

With regard to reports, a simple method has been provided to direct report data into files, through filters or into the spooler system. One simply has to edit the '/etc/prntprew' file to one's own specifications. The default is through the Enhanced Printer Spooler to the 'spr' device.

An error was geoerated when the detail report was specified. An examinatioo of the documentation revealed that the report was referred to by two different names in the glossary. Changing the name from 'detaillist' to 'detailreport' corrected the situation.

The system account numbers range (in three digit numbers) from 100 through 999 as follows in the default settings:

Balance Sheet Accounts Range Default

100 - 149 Current Assets
150 - 179 Fixed Assets
180 - 199 Other Assets
200 - 249 Current Liabilities
250 - 299 Long-term Debt
300 - 399 Capital

Income Statement Accounts

400 - 599 Income
600 - 699 Operating Expense
700 - 799 General & Admin. Expense
800 - 999 Other Income

Where three digit account numbers may not seem to be enough for your company, a four digit system is available. The default ranges simply add one '0' onto the value above. Thus Current Assets would range from 1000 through 1499 if you opt for the extended version.

In addition to the account ranges, a two (four in the extended version) digit category code is provided to allow for subsidiary accounts or divisional accounting. Account information may be calculated and displayed/printed on all categories or simply one category in any report.

Multiple journals are supported in an alphabetic format. Headings and two digit descriptors are entered through the maintenance programs. It is recommended that 'CD' be used to represent cash disbursements and so on. This allows for handy referente during entry without having to refer to lists.

The system is batch oriented single entry. Five digit reference numbers are provided for, and each batch is user-assigned a unique number. Multiple entries to any batch are allowed and batch totals are reported at the end of a session.

When entering data to a batch, one specifies the accounting period (01 to 12), the journal and batch number desired. In case one needs to know which journals have been used in the current accounting period, a help facility is provided. One item of note is that the system did not appear to recognize a numeric journal which had been used in the period. It is my opinion that if the system disallows numeric journal keys, then these should not be allowed at the maintenance level. Thus the system supports numeric journal types for all functions except the batch entry help mode.

An abundance of help is available at most steps of data entry. The help function is usually the entry of an '?' at the entry level prompt. The use of several special characters allow for the auto-entry of repetitious data such as descriptions and reference numbers.

Upon completion of the g-l entry sequence, a request is made as to whether or not you desire a hard copy dump of the entered data (for backup purposes). This step allows for the skipping of a batch dump in a minor batch.

A status report allows for the selective dumping to screen (and optionally printer) of the entries of any (or all) categories of any (or all) periods of any single account. Account totals are displayed at the end of the report. The format of this report is exactly the same as the batch backup dump described above. Journal descriptors are in the form 'AABBCC' where 'AA' is the period, 'BB' is the journal and 'CC' is the batch number of an entry. The sequence number, customer reference number, date, descriptton and amount is displayed as well.

Two programs allow for the P&L, balance statement and financial statement determinations to be made on a year to date, or month to date basis. These report setup programs allow for free typing a descriptive line on each report.

The reports generated are clean, clear and concise. They are reported in standard financial fashion and require absolutely no explanation. No sample reports need be reproduced here.

No account totals are kept by the system. All totals are calculated on a 'month-to-date' or 'year-to-date' basis as required. Thus it is a simple task to adjust batches for missing (forgotten) entries or to correct totals and regenerate reports.

One simple problem appears to be present in the system. There appears to be no way to generate a year-end rollover for the financial figures and a clearout of detail data for quarterly comparisons or year to year analysis. The only thing I have been able to figure is that one must clear out all batches for a period when entering that accounting period, or to painstakingly remove all details from the files, entry by entry at the start of a new year.

Having spoken to the persons in question about this 'year-end' situation, I have been informed that the method is to rename the data and key files for the batches (or to copy them off the drive) and then start again from scratch for a new year. This is quite elegant, but really should have been documented.

In summary, the system is extremely simple to use and no apparent major defects exist in the system. My only request would be that the manual be more complete and document clearly the pitfalls one would expect in installing the system.

A quality rating of A- is given to this system.

SSB DOS

UTILITIES W/FLEX EQUATES

FLEX Equates

The listing of FLEX equates contains most of the storage locations, DOS user callable subroutines, and various dummy data structures and equates needed for proper 6809 assembly language programming. All equated values were taken from the TSC FLEX Programmer's Manual for the 6809 version of FLEX.

Any of you out there who have programmed in IB 360/370 assembler know the use of a DSECT (Du my Section). I have defined Dsects for an FCB (File Control Block) and a SIR (System Information Record). The format described by a Dsect may be associated with a particular area of storage. For example, to access the various fields within an FCB, an index register should contain the address of an FCB storage area. It is then just a matter of using the variables in the FCB Dsect, along with the index register, to access any field in the area.

```
Example:
        LDX   #SYSFCB   X-> FCB storage area
        LDD     #SIRTS    point to System Info
Record
        STD   FCBCP,X   set trk/sec in FCB
        LDA   #XRSS     get function code to read
        STA   FCBFC,X   save code in FCB
        JSR   FMSCAL    read the SIR from disk
        BCS   ERROR     branch if error
        LDY     #SIR FCB+FCBSB  point to SIR's
sector buffer
        LDD   SIRVOL,Y  get volume# of disk
              .
              .
              .
```

EXTEND a FLEX Directory

Those of you who use FLEX know that when a disk is initially formatted, track 0 contains the directory sectors starting at sector 5 and ending at the last sector of track 0; sector A. This gives you six sectors of directory entries, and at ten entries per sector, this yields 60 entries.

Well, if you have single sided, single density disk drives, 60 files on one disk will probably do you just fine. Once you start advancing to double sided, double density and even double track drives, the small initial directory size becomes noticable. I have two of these "octo-density" type drives and it is not uncommon for me to have more than 60 files on my 2000 sector capacity diskette. Sure, FLEX will extend the directory automatically after the initial directory sectors are used up, but the additional sectors are taken from the first available free sector, and thus the directory becomes fragmented across the disk. Furthermore, FLEX only extends by one sector at a time, so you really start to notice the extra seeking needed to find these fragmented file entries. The optimum solution is to allocate a large enough directory space when the disk is for atted. The directory will be contiguous and files will be found much faster.

I'm sure a lot of you realize this problem and have been to busy (or too lazy) to write you own utility to solve the problem. Using some kind of repair utility to change the directory links by hand does the job, but is a nuisance and can be disasterous if it is not done carefully.

The EXTEND command was written to solve this very problem. It is designed to be used following a disk format, and will increase the initial six sector directory by 1 to 30 sectors, thus yielding a possible additional 300 directory file entries. This maximum extension value may be changed of course, to suit the individual. Since the program finds the last directory node by chaining through the directory sectors, this command will work with any type of disk (single/double sided, single/double density, 5"/8", etc.), and can be used at other than disk format efficiency of directory searching, use after a disk format. Syntax is described in the program listing, but here are a couple of examples:

EXTEND E=20,D=1

Extend the directory of the disk in drive 1, by 20 sectors (disk will have 60+200=260 contiguous directory entries initially).

EXTEND

Extend the directory of the disk in the work drive by 10 sectors.

FLEX Setime Utility

The following SETIME routine is almost an exact copy of SSB's DOS setime routines, except that it has been modified to operate under FLEX (since FLEX does not provide a routine to set the time for those of you who have the MM58167 on-board clock).

In addition to updating the real time clock, SETIME also modifies the FLEX date register appropriately, so that the DATE command need not be used separately to set the month, day and year. One nice feature added to SETIME is the ability to set the year, and because it is a parameter on the command, it can be put in your startup file. By utilizing the FLEX date register to store the year (as the DATE command does), there is no need to have it hard-coded in the SETIME command. The format of the command is:

SETIME

The program will prompt you to enter the current date and/or time

SETIME D

This will give you the current time and date

SETIME 83

Change the year in FLEX's date register to "83"

SETIME 83 D

Set the year to "83" and display the date/time while you're at it

The real time clock for my 6809 system is at $F700 (label CLOCK as defined in the FLEX equates). Change this value appropriately.

For a small fee of $12.00 (U.S), I will send you the source to the SETIME and EXTEND commands, as well as the FLEX equates, on a 5" floppy disk. Please specify whether you want it on 40 tracks or 80 tracks. Price includes cost of disk. Make check or moneyorder payable to:

Scott Fraser
547 Sharron Bay
Winnipeg, Manitoba, Canada
R2G 0H8

```
                *
                * FLEX Subroutine Linkages
CD00 FLEX       EQU     $CD00
CD00 COLDS      EQU     FLEX+$00    coldstart entry point
CD03 WARMS      EQU     FLEX+$03    warmstart entry point
CD06 RENTER     EQU     FLEX+$06    DOS main loop reentry point
CD09 INCH       EQU     FLEX+$09    input character
CD0C INCH2      EQU     FLEX+$0C    input character
CD0F OUTCH      EQU     FLEX+$0F    output character
CD12 OUTCH2     EQU     FLEX+$12    output character
CD15 GETCHR     EQU     FLEX+$15    get character
CD18 PUTCHR     EQU     FLEX+$18    put character
CD1B INBUF      EQU     FLEX+$1B    input into line buffer
CD1E PSTRNG     EQU     FLEX+$1E    print string with CR
CD21 CLASS      EQU     FLEX+$21    classify character
CD24 PCRLF      EQU     FLEX+$24    print CR and LF
CD27 NXTCH      EQU     FLEX+$27    get next buffer character
CD2A RSTRIO     EQU     FLEX+$2A    restore I/O vectors
CD2D GETFIL     EQU     FLEX+$2D    get file specification
CD33 LOAD       EQU     FLEX+$30    file loader
CD33 SETEXT     EQU     FLEX+$33    set extension
0000 BIN        EQU     0           "
0001 TXT        EQU     1           "
0002 CMD        EQU     2           "
0003 BAS        EQU     3           "
0004 SYS        EQU     4           "
0005 BAK        EQU     5           "
0006 $OR        EQU     6           "
0007 DAT        EQU     7           "
0008 RAC        EQU     8           "
0009 DIR        EQU     9           "
000A PRT        EQU     10          "
000B OUT        EQU     11          "
CD36 ADDBYA     EQU     FLEX+$36    add B-register to X-register
CD39 OUTDEC     EQU     FLEX+$39    output decimal number
CD3C OUTHEX     EQU     FLEX+$3C    output hexadecimal number
CD3F RPTERR     EQU     FLEX+$3F    report error
CD42 GETHEX     EQU     FLEX+$42    get a hexadecimal number
CD45 OUTADR     EQU     FLEX+$45    output hexadecimal address
CD48 INDEC      EQU     FLEX+$48    output decimal number
CD4B CMDCMD     EQU     FLEX+$4B    call DOS as a subroutine
CD4E STAT       EQU     FLEX+$4E    check terminal input status
                *
                * File Management System Entry Points
                *
CB40 SYSFCB     EQU     $CB40       System FCB
0400 FMS        EQU     $0400       File Management System entry
0400 FMSINT     EQU     FMS+$00     FMS initialization
0403 FMSCLS     EQU     FMS+$03     FMS close
0406 FMSCAL     EQU     FMS+$06     FMS call
                *
                * Global Variables
                *
0409 FCBASE     EQU     FMS+$09     FCB base pointer
040B FCBCUR     EQU     FMS+$0B     current FCB address
0435 FCBVER     EQU     FMS+$35     verify flag
                *
                * DOS memory map
                *
CC80 LINEBUF    EQU     $CC80       80 (SCOFF (128 byte line buf)
CC00 MAP        EQU     $CC00       start of map
CC00 BS         EQU     MAP+$00     TTYSET backspace char
CC01 DEL        EQU     MAP+$01     TTYSET delete character
CC02 EOL        EQU     MAP+$02     TTYSET end of line character
CC03 DEPTH      EQU     MAP+$03     TTYSET depth count
CC04 WIDTH      EQU     MAP+$04     TTYSET width count
CC05 NULL       EQU     MAP+$05     TTYSET null count
CC06 TAB        EQU     MAP+$06     TTYSET tab character
CC07 BSE        EQU     MAP+$07     TTYSET backspace echo character
CC08 EJECT      EQU     MAP+$08     TTYSET eject count
CC09 PAU        EQU     MAP+$09     TTYSET pause control
CC0A ESC        EQU     MAP+ 0A     TTYSET escape character
CC0B SYDRV      EQU     MAP+$0B     system drive number
CC0C WKDRV      EQU     MAP+$0C     work drive number
CC0D SYSCR1     EQU     MAP+$0D     system scratch
CC0E SYDR       EQU     MAP+$0E     system date registers
CC11 LSTBM      EQU     MAP+$11     last terminator
CC12 UCTA       EQU     MAP+$12     user command table address
CC14 BUFPNT     EQU     MAP+$14     line buffer pointer
CC16 ESCRR      EQU     MAP+$16     escape return register
CC18 CURC       EQU     MAP+$18     current character
CC19 PREVC      EQU     MAP+$19     previous character
CC1A CLN        EQU     MAP+$1A     current line number
CC1B LAD        EQU     MAP+$1B     loader address offset
CC1D TRFLG      EQU     MAP+$1D     transfer flag
CC1E TRADR      EQU     MAP+$1E     transfer address
CC20 FMSERR     EQU     MAP+$20     error type
CC21 IOFLG      EQU     MAP+$21     special I/O flag
CC22 OSWTCH     EQU     MAP+$22     output switch
CC23 ISWTCH     EQU     MAP+$23     input switch
CC24 FOA        EQU     MAP+$24     file output address
CC26 FIA        EQU     MAP+$26     file input address
CC28 CMDFLG     EQU     MAP+$28     command flag
CC29 CBC        EQU     MAP+$29     current output column
CC2A SYSCH2     EQU     MAP+$2A     system scratch
CC2B MEMEND     EQU     MAP+$2B     memory end
CC2D ERV        EQU     MAP+$2D     error name vector
CC2F FIEF       EQU     MAP+$2F     file input echo flag
CC30 SYSCR3     EQU     MAP+$30     system scratch
CC4E SYSCON     EQU     MAP+$4E     system constants
CCC0 PRINIT     EQU     MAP+$C0     printer initialize
CCDB PRCHK      EQU     MAP+$DB     printer ready check
CCE4 POUT       EQU     MAP+$E4     printer output

CCF8 SYSCR4     EQU     MAP+$F8     system scratch

                *
                * Dsect for an FCB
                *
0000                    ORG     $0000
0000            FCBFC   RMB     1           function code
0001            FCBESB  RMB     1           error status byte
0002            FCBAS   RMB     1           security status
                0001    ASREAD  EQU     1       **open for read
                0002    ASWRIT  EQU     2       **open for write
0003            FCBDN   RMB     1           drive number
0004            FCBNAM  RMB     8           file name
000C                    RMB     3           extension
000F            FCBFA   RMB     1           file attributes
                0080    FAWP    EQU     $10000000   **write protect
                0040    FADP    EQU     $01000000   **delete protect
                0020    FARP    EQU     $00100000   **read protect
                0010    FACP    EQU     $00010000   ** catalog protect
0010            FCBRS1  RMB     1           reserved for future use
0011            FCBSDA  RMB     2           starting disk addr of file
0013            FCBEDA  RMB     2           ending disk addr of file
0015            FCBFS   RMB     2           file size
0017            FCBFSM  RMB     1           file sector map indicator
                0000    FSMSEQ  EQU     0       **sequential file
                0002    FSMRAN  EQU     2       **random file
0018            FCBRS2  RMB     1           reserved for future use
                0019    FCBFCD  EQU     *       file creation date
0019            FCDMTH  RMB     1           **month
001A            FCDDAY  RMB     1           **day
001B            FCDYR   RMB     1           **year
001C            FCBLP   RMB     2           FCB list pointer
001E            FCBCP   RMB     2           trk/sec currently in sec buff
0020            FCBCRN  RMB     2           current record number
0022            FCBDI   RMB     1           data index
0023            FCBRI   RMB     1           random index
0024            FCBNWB  RMB     11          name work buffer
002F            FCBCDA  RMB     3           current directory address
0032            FCBFDD  RMB     3           first deleted dir ptr
0035            FCBSCR  RMB     11          scratch bytes
0038                    ORG     FCBSCR+6
0038            FCBSCF  RMB     1           space compression flag
```

```
                0000  SCFSC   EQU   $00       **perform space compr.
                00FF  SCFNSC  EQU   $FF       **perform no space compr.
0040                  ORG     FCBSCR+11
                0040  FCBSB   FCB   $         sector buffer
0080                  SBLINK  RMB   2         next trk/sector in chain
0042                  SRSI    RMB   2         reserved for future use
0044                  SRDATA  RMB   252       data storage
                0140  FCBLEN  EQU   *         length of FCB
                *
                * Function Codes
                *
0000  GRAB    EQU   0         read/write next byte/char
0001  XOPEAD  EQU   1         open for read
0002  XOWRIT  EQU   2         open for write
0003  XOUPDT  EQU   3         open for update
0004  XCLOSE  EQU   4         close file
0005  XRWIND  EQU   5         rewind file
0006  XOPDIR  EQU   6         open directory
0007  XGTIR   EQU   7         get information record
0008  XPTIR   EQU   8         put information record
0009  XRSS    EQU   9         read single sector
000A  XWSS    EQU   10        write single sector
000B  XRESI   EQU   11        reserved for future use
000C  XDELET  EQU   12        delete file
000D  XRENAM  EQU   13        rename file
000E  XRES2   EQU   14        reserved for future use
000F  XNSS    EQU   15        next sequential sector
0010  XOSIR   EQU   16        open system info rec
0011  XGRB    EQU   17        get random byte from sector
0012  XPRB    EQU   18        put random byte in sector
0013  XRES3   EQU   19        reserved for future use
0014  XFIND   EQU   20        find next drive
0015  XPOSN   EQU   21        position to record n
0016  XBOR    EQU   22        backup one record
                *
                * Dsect for a SIR
                *
0000                  ORG     $0000
0000                  RMB     16        16 byte header
0010  SIANAM  RMB     8         volume name
0018                  RMB     3         extension
001B  SIMVOL  RMB     2         volume number
001D  SIRFSB  RMB     2         beginning of free chain
001F  SIRFSE  RMB     2         end of free chain
0021  SIRFSS  RMB     2         # sectors in free chain
                0023  SIRCRE  EQU   *         creation date of disk
0021  SIRMTH  RMB     1         **month
0024  SIRDAY  RMB     1         **day
0025  SIRYR   RMB     1         **year
0026  SIRMTS  RMB     2         maximum trk/sec available
                0028  SIRLEN  EQU   *         SIR length
                *
                * Miscellaneous equates
                *
0003  SIRTS   EQU   $0003    trk/sec of SIR
0009  DIRTS   EQU   $0009    trk/sec of 1st node in dir
                *
0004  EOT     EQU   4         end of text delimiter
000A  CRLF    EQU   $000A     carriage return, line feed
000D  CR      EQU   $0D       carriage return
000A  LF      EQU   $0A       line feed
0007  BELL    EQU   $07       bell
0020  SP      EQU   $20       space
                *
0000  URAM    EQU   $0000     to $BFFF (User RAM area)
C000  STKA    EQU   $C000     to $C07F (SP limited to C07F)
C100  UCA     EQU   $C100     to $C6FF (Util Cmd Area)
C700  SPS     EQU   $C700     to $C87F (SysInit & Spooler)
C980  SFA     EQU   $C980     to C9FF (System Files Area)
CC00  DOS     EQU   $CC00     to D3FF (DOS)
F700  CLOCK   EQU   $F700     real time clock (DOS)
DE00  DDRV    EQU   $DE00     to DFFF (Disk Drivers)
                      OPT   LIS

0 ERROR(S) DETECTED
```

```
                *
                * Burrage/Fraser Software Consultants
                * 1 Pleasant Bay
                * Winnipeg, Manitoba, Canada
                * R2K 0C9
                * July, 1982
                *
                * This EXTEND command takes a newly formatted disk
                * and adds several more sectors to the directory.
                * The maximum allowable amount to extend by is 30
                * sectors (yields an additional 300 directory entries).
                *
                * It is called as:  EXTEND 10=drv#1],E=#secs]
                *
                *   Where:
                *
                *         D=   specifies the drive to extend
                *         E=   specifies the # of sectors to extend by
                *
                * If no parameters are given, then the diskette
                * on the work drive will be extended by
                * 10 sectors (good for 60+100+160 file entries).
                *
                * Note also that since the end of the directory is
                * found by following the linked chain, and that
                * the extension is carried out by following the linked
                * nodes, this command will work with any type of
                * diskette (single/double sided, single/double density,
                * 5"/8", etc).
                *

0000                  ORG     $0000
0000  DIRFCB  RMB     FCBLEN    directory FCB
0140  SIRFCB  RMB     FCBLEN    System Information Record FCB
C100                  ORG     UCA
C100 20  08   START   BRA     START1
C102 02               FCB     2         version 2
                000A  DEFSEC  EQU   10      default # sectors to extend by
                0001  MAXDRV  EQU   1       maximum drive number
                001E  MAXXTD  EQU   30      max # of sectors to extend by
C103                  DRV     RMB   1       holds drive number
C104 00               DIRSEC  FCB   0       for 16 bit subtract
C105                  SECTRS  RMB   1       holds sector extension amt
C106                  TRKSEC  RMB   2       holds trk/sec value
C108                  CTR     RMB   1       counter
C109                  DIREND  RMB   2       holds trk/sec of 1st dir node
                C10B  START1  EQU   *
                *
                * First setup the default drive number and the

EXTEND Flex Directory              2-18-82 TSC ASSEMBLER   PAGE    2

                * default number of sectors to extend the directory
                * by.
                *
C10B B6  CCDC         LDA     WKDRV     default is work drive
C10E B7  C103         STA     DRV       save it
C111 86  0A           LDA     #DEFSEC   default # of sectors
C113 B7  C105         STA     SECTRS    save it
                *
                * Now parse for the input parameters
                *
C116 BD  C204         JSR     CMDPRM    get parms
C119 1025 0009        LBCS    ERR01     bad parm -> error
```

```
C11D B6  C103         LDA     DRV       get drive # to print
C120 8A  30           ORA     #'0       convert to ascii
C122 B7  C532         STA     OUTDRV
C125 8E  C514         LDX     #DRVMSG   print msg before starting
C128 BD  CD1E         JSR     PSTRNG    print msg
C12B BD  CD09         JSR     INCH      and get response
C12E 84  5F           ANDA    #$5F      convert to upper case
C130 81  59           CMPA    #'Y       if yes then continue on
C132 1026 00AD        LBNE    EX005     if anything else then exit
                *
                * Set drive in FCB's
                *
C136 8E  0000         LDX     #DIRFCB   point to directory FCB
C139 B6  C103         LDA     DRV       get drive number
C13C A7  03           STA     FCBDRV,X  save in directory FCB
C13E 8E  0140         LDX     #SIRFCB   point to SIR FCB
C141 A7  03           STA     FCBDRV,X  save in SIR FCB
C143 CC  0003         LDD     #SIRTS    point to SIR
C146 BD  C252         JSR     READ1     get SIR
C149 1079 00ME        LBCS    ERR02     report error
                *
                * The system information record has been read.
                * Extend the directory by the number of sectors
                * specified.  To do this, the last sector of the
                * directory must be pointed to the first sector in the
                * free chain.  The link of the last extended sector
                * in the directory must be zeroed.  The SIR must
                * then have its free chain pointer updated, as well
                * as its "number of free sectors" value.
                *
C140 8E  0000         LDX     #DIRFCB   point to directory FCB
C190 CC  0009         LDD     #DIRTS    point to beginning of dir
                C153  EX010   EQU     *
C153 FD  C108         STD     DIREND    save link
C156 BD  C252         JSR     READ1     get sector
C159 1079 00ME        LBCS    ERR02     report error
C15D EC  88 40        LDD     SBLINK,X  get link
C160 26  F1           BNE     EX010     if <>0 then continue search
                *
                * Found the end of the directory chain.
                *
                * Update this to point to the first sector of the
                * free chain
                *
C162 108E 0180        LDY     #SIRFCB+FCBSB point to SIR's sec buf
C166 EC  A8 1D        LDD     SIRFSB,Y  get first free sector
C169 ED  88 40        STD     SBLINK,X  and save in directory node
                *
                * write out updated dir sector
                *
C16C 8E  0000         LDX     #DIRFCB
C16F FC  C109         LDD     DIREND
C172 BD  C25C         JSR     WRITE1    write back this entry
C175 25  7C           BCS     ERR03     report error
                *
                * Now chain through the free space chain "sectrs"
                * times.  Once at end, zero its link
                *
C177 B6  C105         LDA     SECTRS    # sectors to extend by
C17A B7  C108         STA     CTR       save away
C17D EC  A8 1D        LDD     SIRFSB,Y  get ptr to first free link
C180 8E  0000         LDX     #DIRFCB   point to an FCB
                C183  EX011   EQU     *
C183 FD  C106         STD     TRKSEC    save this link
C186 BD  C252         JSR     READ1     read a sector
C189 25  60           BCS     ERR02     report error
                *
                * Before going on to the next node,
                * zero out this one's data area
                * and write back on disk (ensures
                * a "clean" directory)
                *
C18B A6  FE           LDA     #$FE      # bytes of data to zero
C18D 30  02           LEAX    $02,X     point to area
                C190  EX013   EQU     *
C190 6F  80           CLR     0,X+      zero a byte
C192 4A               DECA
C193 26  FB           BNE     EX013     continue until done
C195 8E  0000         LDX     #DIRFCB   point back to FCB
C198 7A  C108         DEC     CTR       extended enough?
C19B 27  0F           BEQ     EX012     yes, then quit search
C19D FC  C106         LDD     TRKSEC    recall its trk/sec
C1A0 BD  C25C         JSR     WRITE1    write back to disk
C1A3 1025 0008        LBCS    WRERR     on error quit
C1A7 EC  88 40        LDD     SBLINK,X  nop, get next link
C1AA 26  D7           BNE     EX011     keep going if not zero
                C1AC  EX012   EQU     *
C1AC EC  88 40        LDD     SBLINK,X  start of free space
C1AF 27  4A           BEQ     ERR04     if no room left then error!
C1B1 ED  A8 1D        STD     SIRFSB,Y  save in SIR
C1B4 EC  A8 21        LDD     SIRFSS,Y  get size of free chain
C1B7 B3  C104         SUBD    DIRSEC    sub off extension size
C1BA ED  A8 21        STD     SIRFSS,Y  save back to SIR
                *
                * Write out the last directory node
                *
                * (in DIRFCB) but zero its linkage field.
                *
C1BD CC  0000         LDD     #$0000    zero link field
C1C0 ED  88 40        STD     SBLINK,X
                *
                * write updated sector
                *
C1C3 FC  C106         LDD     TRKSEC    get trk/sec to write
C1C6 BD  C25C         JSR     WRITE1
C1C9 25  38           BCS     ERR03     report error
                *
                * Write out updated SIR
                *
C1CB 8E  0140         LDX     #SIRFCB   point to SIR FCB
C1CE CC  0003         LDD     #SIRTS    point to SIR
C1D1 BD  C25C         JSR     WRITE1    write it out
C1D4 25  10           BCS     ERR03     report error
                *
                * All done
                *
C1D6 8E  C207         LDX     #DONE     done msg
C1D9 BD  CD1E         JSR     PSTRNG    print it
C1DC 8E  C104         LDX     #DIRSEC   print out # secs extended
C1DF 5F               CLRB              suppress leading zeroes
C1E0 BD  CD39         JSR     OUTDEC    print number
                C1E3  EX009   EQU     *
C1E3 7E  CD03         JMP     WARMS     return to flex
                *
                * Error routines
                *
                C1E6  ERR01   EQU     *
C1E6 8E  C288         LDX     #BADPRM   get bad parameter msg
C1E9 20  13           BRA     ERR       print and return
                C1EB  ERR02   EQU     *
C1EB BD  CD3F         JSR     RPTERR    report error first
C1EE 8E  C274         LDX     #RDERR    get read error msg
C1F1 20  0B           BRA     ERR       print and return
                C1F3  ERR03   EQU     *
C1F3 BD  CD3F         JSR     RPTERR    report error first
C1F6 8E  C27F         LDX     #WRERR    get write error msg
C1F9 20  03           BRA     ERR       print msg and return
                C1FB  ERR04   EQU     *
C1FB 8E  C288         LDX     #TOOBIG   get dir too big msg
                C1FE  ERR     EQU     *
C1FE BD  CD1E         JSR     PSTRNG    print msg
C201 7E  CD03         JMP     WARMS     return to FLEX
```

```
                        *
                        * Name      - CHKPRM
                        * Function - This routine extracts the optional
                        *            parameters from the line buffer for
                        *            the EXTEND cmd, and saves the values
                        *            in their appropriate places
                        *
                        * On Exit -> If a bad parm was found, then carry is
                        *            set, otherwise clear
                        *
                        * All registers are preserved
                        *
                  C204  CHKPRM  EQU   *
C204 34   36            PSHS  X,Y,D       save regs
C206 86   03            LDA   #3          Number of possible parms
C208 B7   C108          STA   CTR         save in temp
                  C20B  CHK003  EQU   *
C20B BD   CD27          JSR   NXTCH       get a char
C20E 81   0D            CMPA  #CR         a cr?
C210 27   37            BEQ   CHKOK       yes, then done
C212 81   44            CMPA  #'D         is this a drive spec?
C214 26   16            BNE   CHK001      nop, try next parm
C216 BD   CD27          JSR   NXTCH       skip over sign
C219 BD   CD42          JSR   GETHEX      get drive number
C21C 5D                 TSTB              check if number there
C21D 27   25            BEQ   CHK002      nop
C21F 1F   10            TFR   X,D         get drive #
C221 1083 0001          CMPD  #MAXDRV     valid range?
C225 22   27            BHI   CHKBAD      Nop
C227 F7   C1D3          STB   DRV         save drive number
C22A 20   18            BRA   CHK002      get next parm
                  C22C  CHK001  EQU   *
C22C 81   45            CMPA  #'E         Is this the extend parm?
C22E 26   1E            BNE   CHKBAD      nop, then unknown option
C230 BD   CD27          JSR   NXTCH       skip over sign
C233 BD   CD48          JSR   INDEC       get # sectors to extend
C236 5D                 TSTB              check for # there
C237 27   0B            BEQ   CHK002      nop, find next parm
C239 1F   10            TFR   X,D         get # sectors
C23B 1083 001E          CMPD  #MAXXTD     valid range?
C23F 22   0D            BHI   CHKBAD      Nop
C241 F7   C1D5          STB   SECTRS      save value
                  C244  CHK002  EQU   *
C244 7A   C108          DEC   CTR
C247 26   C2            BNE   CHK003      continue parm check
                  C249  CHKOK   EQU   *
C249 1C   FE            CLC               set good RC
C24B 35   36            CHKRET PULS X,Y,D   restore registers
C24D 39                 RTS               return
                  C24E  CHKBAD  EQU   *
C24E 1A   01            SEC               set bad RC
C250 20   F9            BRA   CHKRET      return

EXTEND Flex Directory                2-18-82 TSC ASSEMBLER   PAGE    6

                        *
                        * Name      - READ1
                        * Function- This routine reads in the track/sector
                        *            of the disk in the drive specified by an
                        *            FCB pointed to by the X ixr.  Reg D
                        *            specifies the trk/sec to read in
                        *
                        * On Exit -> the carry is set if a read error,
                        *            otherwise clear
                        *
                        * All registers except A are preserved
                        *
                  C252  READ1   EQU   *
C252 ED   88 1E         STD   FCHCP,X     set trk/sec
C255 86   09            LDA   #XRSS       get function code
C257 A7   84            STA   FCBFC,X     set code in FCB
C259 7E   D406          JMP   FMSCAL      read and return
                        *
                        * Name      - WRITE1
                        * Function - This routine writes a trk/sec given in
                        *            reg D to the drive described by an FCB
                        *            pointed to by the X ixr.
                        *
                        * On Exit -> the carry is set on a write error,
                        *            otherwise clear
                        *
                        * All registers except A are preserved
                        *
                  C25C  WRITE1  EQU   *
C25C ED   88 1E         STD   FCHCP,X     set trk/sec
C25F 86   0A            LDA   #XWSS       set code in FCB
C261 A7   84            STA   FCBFC,X
C263 7E   D406          JMP   FMSCAL      write and return
                        *
                        * Data area for strings
                        *
C266 42 61 64 20        BADPRM  FCC   /Bad Parameter/
C26A 50 61 72 61
C26E 6D 65 74 65
C272 72
C273 04                         FCB   EOT
C274 52 65 61 64        RDERR   FCC   /Read Error/
C278 20 45 72 72
C27C 6F 72
C27E 04                         FCB   EOT
C27F 57 72 69 74        WRERR   FCC   /Write Error/
C283 65 20 45 72
C287 72 6F 72
C28A 04                         FCB   EOT
C28B 44 69 72 65        TOOBIG  FCC   /Directory takes up all of disk/
C28F 63 74 6F 72
C293 79 20 74 61
C297 6B 65 73 20
C29B 75 70 20 61
C29F 6C 6C 20 6F
C2A3 66 20 64 69
C2A7 73 6B
C2A9 000A                       FDB   CRLF
C2AB 50 6C 65 61        NOTEMP  FCC   /Please reformat disk and try again!/
C2AF 73 65 20 72
C2B3 65 66 6F 72
C2B7 6D 61 74 20
C2BB 64 69 73 6B
C2BF 20 61 6E 64
C2C3 20 74 72 79
C2C7 20 61 67 61
C2CB 69 6E 21
C2CE 04                         FCB   EOT
C2CF 000A               DONE    FDB   CRLF
C2D1 45 78 74 65                FCC   /Extension Completed Successfully/
C2D5 6E 73 69 6F
C2D9 6E 20 43 6F
C2DD 6D 70 6C 65
C2E1 74 65 64 20
C2E5 53 75 63 63
C2E9 65 73 73 66
C2ED 75 6C 6C 79
C2F1 000A                       FDB   CRLF
C2F3 4E 75 60 62        NUMSEC  FCC   /Number of Sectors Extended by = /
C2F7 65 72 20 6F
C2FB 66 20 53 65
C2FF 63 74 6F 72
C303 73 20 65 78
C307 74 65 6E 64
C30B 65 64 20 62
C30F 79 20 30 20
C313 04                         FCB   EOT
C314 45 78 74 65        RDYMSG  FCC   /Extending directory on drive #/
C318 6E 64 69 6E
C31C 67 20 64 69
C320 72 65 63 74
C324 6F 72 79 20
C328 6F 6E 20 64
C32C 72 69 76 65
C330 20 23
C332                    OUTDRY  RMB   1
C333 2C 20 61 72                FCC   ", are you ready (Y/N)? "
C337 65 20 79 6F
C33B 75 20 72 65
C33F 61 64 79 20
C343 28 59 2F 4E
C347 29 3F 20
C34A 04                         FCB   EOT
                                END   START

0 ERROR(S) DETECTED

                        *
                        * Setime - The SET TIME/DATE transient allows the
                        *          user of SSB's Real Time Clock (RTC) to set
                        *          the time, date and year of the hardware
                        *          device.
                        *
                        * It is called as:
                        *
                        *          SETIME       (set time and date)
                        *
                        *          SETIME D     (display time & date)
                        *
                        *          SETIME YY    (set year given by "YY"
                        *                        digits)
                        *
                        *          SETIME YY D  (set year and display
                        *                        time & date)
                        *
                        *   NOTE: "YY" should be between 00 and 99
                        *
                        * This program does not use interrupts to update
                        * the time and date string on the display
                        * once a second; instead uses a software loop
                        * to determine a 1 cycle/sec interval
                        *
                        * This is a slight modification of DOS' SETIME cmd
                        * to work under FLEX (Sept, 1982)
                        *
                  C387  ZROTIN  EQU   RDTIME      "read time" rtn
                  FB0C  PDATA   EQU   $FB0C       addr of PDATA rtn (SSB MON)
                        *
                        * MM68167 register offsets
                        *
                  0002  SEC     EQU   2           seconds
                  0003  MIN     EQU   3           minutes
                  0004  HOUR    EQU   4           hours
                        *
                  0005  DOW     EQU   5           day of week
                  0006  DOM     EQU   6           day of month
                  0007  MON     EQU   7           month of year

C100                            ORG   UCA
C100 20   23                    BRA   START
                  0002  VN      EQU   2           version 2
C102                    YDATE   RMB   16          holds the date
C112                    YTIME   RMB   16          holds the time
C122                    XTEMP   RMB   3
                  C125  START   EQU   *
C125 10BE F700                  LDY   PCLOCK
C129 BD   CD27                  JSR   NXTCH       get a character
C12C 25   34                    BCS   START1      branch if nothing
C12E 81   44                    CMPA  #'D         just want date/time?
C130 27   22                    BEQ   DISTD       branch if so
                        *
                        * assume a year has been specified
C132 BD   CD21                  JSR   NXTCH       get 1's digit in year
C135 25   28                    BCS   START1      branch if nothing
C137 1F   89                    TFR   A,B         A => B
C139 86   CC19                  LDA   #REVC       get 10's digit
C13C 17   02C9                  LBSR  ASCBIN      convert to binary
C13F 25   4C                    BCS   ERR1        leave if error
C141 B7   CC10                  STA   SYDR+2      save year
C144 BD   CD27                  JSR   NXTCH       anything else?
C147 81   20                    CMPA  #SP         this a space?
C149 26   48                    BNE   XIT         nop, then leave
C14B BD   CD27                  JSR   NXTCH       go past spaces
C14E 25   43                    BCS   XIT         if nothing, leave
C150 81   44                    CMPA  #'D         want date/time?
C152 26   3F                    BNE   XIT         nop, then leave
                  C154  DISTD   EQU   *
C154 BD   CD24                  JSR   PCRLF
C157 BD   CD24                  JSR   PCRLF
C15A 17   010F                  LBSR  DISPLY      yes, print date/time once
C15D BD   CD24                  JSR   PCRLF
C160 20   31                    BRA   XIT         return to FLEX
                  C162  START1  EQU   *
C162 8E   C2EE                  LDX   #DPLMSG
C165 BD   CD1E                  JSR   PSTRNG      print intro
                  C168  START2  EQU   *
C168 17   0101                  LBSR  DISPLY      print time/date
                  C16B  START3  EQU   *
C16B BD   CD4E                  JSR   STAT        check keyboard status
C16E 26   0A                    BNE   START4      branch if char waiting
C170 17   0125                  LBSR  READ
C173 F1   C122                  CMPB  XTEMP       check for 1 second roll over
C176 27   F5                    BEQ   START3      if the time didn't change
C178 20   EE                    BRA   START2
                  C17A  START4  EQU   *
C17A BD   CD09                  JSR   INCH        get waiting character
C17D 84   5F                    ANDA  #$5F        fold lower case to upper
C17F 81   54                    CMPA  #'T         set time?
C181 27   13                    BEQ   SETIME
C183 81   44                    CMPA  #'D         set date?
C185 27   54                    BEQ   SDATE
C187 81   52                    CMPA  #'R         return to FLEX?
C189 26   07                    BNE   START1
C 88 20   06                    BRA   XIT         return to FLEX
                  C18D  ERR1    EQU   *
C18D 8E   C29B                  LDX   #BADYR      get bad year msg
C190 BD   CD1E                  JSR   PSTRNG      print it
                  C193  XIT     EQU   *
C193 7E   CD03                  JMP   WARMS       return to FLEX
                        *
                        * SET THE TIME
                        *
                        *    The input is in the form of:
                        *
                        *          "HH:MM:SS"
                        *
                        *    where "HH", "MM" and "SS" are 2 digit characters
                        *
                        *    "HH" has a maximum range of 0 to 23
                        *    "MM" and "SS" have a maximum range of 0 to 99
                        *
                  C196  SETIME  EQU   *
C196 B6   CD02                  LDA   EOL         get TTY end of line char
C199 34   02                    PSHS  A           and save it
C19B 7F   CD02                  CLR   EOL         set EOL char to null
C19E 8E   C321                  LDX   #TIMST
```

```
C1A6 BD CD1E          JSR   PSTRNG
C1A9 BD CD1B          JSR   INBUF      get time string

C1A7 BD CD42          JSR   GETHEX     get hours digits

C1AA 1F 10            TFR   X,D
C1AC C1 23            CMPB  #$23
C1AE 22 23            BHI   SETIM1
C1B0 E7 24            STB   HOUR,Y

C1B2 BD CD42          JSR   GETHEX     get minute digits
C1B5 25 1C            BCS   SETIM1     if illegal value

C1B7 1F 10            TFR   X,D
C1B9 C1 59            CMPB  #$59
C1BB 22 09            BHI   SETIM1
C1BD E7 22            STB   MIN,Y

C1BF BD CD42          JSR   GETHEX     get second digits
C1C2 25 0F            BCS   SETIM1     if illegal value

C1C4 1F 10            TFR   X,D
C1C6 C1 59            CMPB  #$59
C1C8 22 09            BHI   SETIM1
C1CA E7 22            STB   SEC,Y

C1CC 35 02            PULS  A          restore EOL char
C1CE B7 CC02          STA   EOL

C1D1 20 BF            BRA   START1

              C1D3    SETIM1 EQU  *
C1D3 8E C370          LDX   #ILVMSG
C1D6 BD CD1E          JSR   PSTRNG
C1D9 20 BB            BRA   SETIME     start again
                      *
                      * SET THE DATE
                      *
                      *     The input is in the form of:
                      *
                      *          "DAY,MON,DD[,YYYY]"
                      *
                      *     where "DAY" is a 3 character string for
                      *     the day of the week (SUN - SAT)
                      *
                      *     where "MON" is a 3 character string for
                      *     the month of the year (JAN - DEC)
                      *
                      *     where "DD" is a 2 digit value for the day
                      *     of the month (1 - 31)
                      *
              C1DB    SEDATE EQU  *
C1DB 8E C34F          LDX   #DATST
C1DE BD CD1E          JSR   PSTRNG
C1E1 BD CD1B          JSR   INBUF
C1E4 30 8D 00CB       LEAX  DOWTBL,PCR
C1E8 8D 44            BSR   SEDAT2     convert day of week string
C1EA 25 39            BCS   SEDAT1     if input error
C1EC E7 25            STB   DOW,Y
C1EE 30 8D 0007       LEAX  MONTBL,PCR
C1F2 8D 3A            BSR   SEDAT2     convert month string to bin
C1F4 25 2F            BCS   SEDAT1     if illegal value
C1F6 E7 CC0E          STB   SYDR       save in FLEX date reg
C1F9 C1 09            CMPB  #9         convert to BCD
C1FB 23 02            BLS   *+4
C1FD CB 06            ADDB  #6
C1FF E7 27            STB   MON,Y      save it

C201 BD CD42          JSR   GETHEX     get day of month digit
C204 25 1F            BCS   SEDAT1     if illegal value

C206 1F 10            TFR   X,D
C208 C1 31            CMPB  #$31
C20A 22 10            BHI   SEDAT1
C20C E7 26            STB   DOM,Y
                      *
                      * Convert month to binary and save
                      * in FLEX's date register.
                      *
C20E 34 04            PSHS  B          save hex month
C210 54               LSRB             get ten's digit
C211 54               LSRB
C212 54               LSRB
C213 54               LSRB
C214 86 0A            LDA   #10        multiply by 10
C216 3D               MUL
C217 35 02            PULS  A          get back hex month
C219 84 0F            ANDA  #$00001111 keep 1's digit
C21B 34 02            PSHS  A          save back
C21D EB E0            ADDB  0,S+       add on 10's digit
C21F F7 CC0F          STB   SYDR+1     save in FLEX date reg
C222 16 FF30          LBRA  START1

              C225    SEDAT1 EQU  *
C225 8E C370          LDX   #ILVMSG
C228 BD CD1E          JSR   PSTRNG
C22B 16 FF34          LBRA  START1

              C22E    SEDAT2 EQU  *
C22E C6 03            LDB   #3
C230 CE C122          LDU   #XTEMP

              C233    SEDAT3 EQU  *
C233 BD CD27          JSR   NXTCH      get a character
C236 24 08            BCC   SEDAT4     if alpha-numeric
C238 81 20            CMPA  #SP
C23A 27 F7            BEQ   SEDAT3     eat leading blanks
C23C 81 0D            CMPA  #CR
C23E 27 29            BEQ   SEDAT7
                      *
              C240    SEDAT4 EQU  *
C240 84 5F            ANDA  #$5F       fold lower case to upper
C242 81 41            CMPA  #'A        don't use numeric chars
C244 25 23            BLT   SEDAT7
C246 A7 C0            STA   0,U+
C248 5A               DECB
C249 26 E8            BNE   SEDAT3

C24B BD CD27          JSR   NXTCH      eat terminator character
C24E FE C122          LDU   XTEMP+0    -> get first two chars
C251 86 C124          LDA   XTEMP+2

C254 C6 01            LDB   #1         preset counter

              C256    SEDAT5 EQU  *
C256 11A3 84          CMPU  0,X
C259 26 07            BNE   SEDAT6     try next string
C25B A1 02            CMPA  2,X
C25D 26 03            BNE   SEDAT6     try next string

C25F 1C FE            CLC              indicate success
C261 39               RTS

              C262    SEDAT6 EQU  *
C262 5C               INCB
C263 30 03            LEAX  3,X        try next string
C265 6D 84            TST   0,X        end of table?
C267 26 ED            BNE   SEDAT5

              C269    SEDAT7 EQU  *
C269 1A 01            SEC              indicate failure
C26B 39               RTS
                      *
                      * UPDATE DISPLAY ONCE A SECOND
                      *
              C26C    DISPLY EQU  *
C26C BD C387          JSR   ZRDTIM     update time/date string

C26F 86 0D            LDA   #CR
C271 BD CD0F          JSR   OUTCH

C274 8E C107          LDX   #YDATE
C277 AD 9F F80C       JSR   [PDATA]    print date string

C27B 86 20            LDA   #SP
C27D BD CD0F          JSR   OUTCH
C280 BD CD0F          JSR   OUTCH

C283 8E C112          LDX   #YTIME
C286 AD 9F F80C       JSR   [PDATA]    print time string

C28A 86 20            LDA   #SP
C28C BD CD0F          JSR   OUTCH
C28F BD CD0F          JSR   OUTCH

C292 BD 04            BSR   READ
C294 E7 C122          STB   XTEMP
C297 39               RTS
                      *
                      * READ THE SECONDS REGISTER
                      *
C298 E6 22            READ  LDB   SEC,Y
C29A 39               RTS
                      *
                      * Data Area
                      *
C29B 49 6E 76 61      BADYR FCC   /Invalid YEAR specified!/
C29F 04               FCB   EOT

              C2B3    DOWTBL EQU  *
C2B3 53 55 4E         FCC   /SUN/
C2B6 4D 4F 4E         FCC   /MON/
C2B9 54 55 45         FCC   /TUE/
C2BC 57 45 44         FCC   /WED/

C2BF 54 48 55         FCC   /THU/
C2C2 46 52 49         FCC   /FRI/
C2C5 53 41 54         FCC   /SAT/
C2C8 00               FCB   0          end of table

              C2C9    MONTBL EQU  *
C2C9 4A 41 4E         FCC   /JAN/
C2CC 46 45 42         FCC   /FEB/
C2CF 4D 41 52         FCC   /MAR/
C2D2 41 50 52         FCC   /APR/
C2D5 4D 41 59         FCC   /MAY/
C2D8 4A 55 4E         FCC   /JUN/
C2DB 4A 55 4C         FCC   /JUL/
C2DE 41 55 47         FCC   /AUG/
C2E1 53 45 50         FCC   /SEP/
C2E4 4F 43 54         FCC   /OCT/
C2E7 4E 4F 56         FCC   /NOV/
C2EA 44 45 43         FCC   /DEC/
C2ED 00               FCB   0          end of table

C2EE 000A             DPLMSG FDB  CRLF
C2F0 53 65 74 20      FCC   'Set Time (T), set date (D), '
C30C 72 65 74 75      FCC   'return to FLEX (R)?'
C31E 000A             FDB   CRLF
C320 04               FCB   EOT

C321 000A             TIMST  FDB  CRLF
C323 49 6E 70 75      FCC   'Input time in 24 hour '
C339 66 6F 72 6D      FCC   'format  "23:59:59"'
C34A 00 0A 3E 20      FCB   CR,LF,'>,SP
C34E 04               FCB   EOT

C34F 000A             DATST  FDB  CRLF
C351 49 6E 70 75      FCC   'Input date in "DAY,MON,DD"'
C368 00 0A 3E 20      FCB   CR,LF,'>,SP
C36F 04               FCB   EOT

C370 49 6C 6C 65      ILVMSG FCC  'Illegal input value!!'
C386 04               FCB   EOT
                      *
                      * This routine reads the time and date and stores
                      * the values in YTIME and YDATE respectively
                      *
              0014    STATUS EQU  20         status register
              C387    RDTIME EQU  *
C387 34 76            PSHS  A,B,X,Y,U  save the important stuff
C389 10BE F700        LDY   #CLOCK     -> start of the MM58167 reg
                      *
                      * Start by reading the time
                      *
C38D CE C112          LDU   #YTIME     -> where to put time

C390 86 04            LDA   #HOUR
C392 8D 16            BSR   RDTIM1     convert & store hours

C394 86 03            LDA   #MIN
C396 8D 12            BSR   RDTIM1     convert & store minutes

C398 86 02            LDA   #SEC
C39A 8D 0E            BSR   RDTIM1     convert & store seconds

C39C 86 04            LDA   #EOT
C39E A7 9F            STA   -1,U       set end of string
C3A0 20 38            BRA   RDTIM4     now read the data
                      *
                      * Read the specified register
                      *
              C3A2    XREAD  EQU  *
C3A2 E6 A6            LDB   A,Y
C3A4 6D A8 14         TST   STATUS,Y   test for register roll over
C3A7 26 F9            BNE   XREAD
C3A9 39               RTS
                      *
                      * Convert BCD digits to ASCII BCD
                      *
              C3AA    RDTIM1 EQU  *
C3AA 8D F6            BSR   XREAD      read the specified register
C3AC 1F 98            TFR   B,A
                      *
                      * Convert MSB
                      *
C3AE 44               LSRA
C3AF 44               LSRA
C3B0 44               LSRA
C3B1 44               LSRA
                      *
                      * Convert LSB
                      *
C3B2 C4 0F            ANDB  #$F
                      *
                      * Convert MSB & LSB to ASCII
                      *
C3B4 C3 3030          ADDD  #'0*256+'0
C3B7 ED C1            STD   0,U++      store characters

C3B9 86 3A            LDA   #':
C3BB A7 C0            STA   0,U+

C3BD 39               RTS
                      *
                      * Convert binary number to corresponding string
                      *
              C3BE    RDTIM2 EQU  *
C3BE 8D E2            BSR   XREAD      read the specified register
C3C0 C1 09            CMPB  #9         check BCD range (> 9)?
```

```
C3C2 23 02              BLS     *+4
C3C4 2D 06              SUBB    #6              make it binary
C3C6 34 04              PSHS    B
C3C8 58                 ASLB
C3C9 EB E0              ADDB    0,S+
C3CB 3A                 ABX                     -> pickup string

C3CC C6 03              LDB     #3              move 3 characters

                C3CE    RDTIM3  EQU     *
C3CE A6 80              LDA     0,X+            move table char to ...
C3D0 A7 C0              STA     0,U+            parameter string area
C3D2 5A                 DECB
C3D3 26 F9              BNE     RDTIM3

C3D5 86 20              LDA     #SP
C3D7 A7 C0              STA     0,U+            install seperator

C3D9 39                 RTS
                *
                * Read the data
                *
                C3DA    RDTIM4  EQU     *
C3DA CE C1D2                    LDU     #YDATE  -> where to put date

C3DD 30 8D FECF         LEAX    DOWTBL-3,PCR -> day of week table
C3E1 86 05              LDA     #DOW            read day of week
C3E3 8D 09              BSR     RDTIM2          convert to ascii string

C3E5 30 8D FED0         LEAX    MONTBL-3,PCR -> month table
C3E9 86 07              LDA     #MON            read month
C3EB 8D 01              BSR     RDTIM2          convert to ascii string

C3ED 86 06              LDA     #DOM            read day of month
C3EF 8D 89              BSR     RDTIME          convert to ascii BCD

C3F1 86 20              LDA     #SP
C3F3 A7 5F              STA     -1,U
                *
                * The following code stores the year portion of
                * the date in the YDATE string. The last 2
                * digits in the year are gotten from the
                * FLEX date register and converted to a 2
                * digit ascii value.
                *
C3F5 CC 3139            LDD     #'1*256+'9 "19"
C3F8 ED C1              STD     0,U++
C3FA A6 C210            LDA     $YDATE+2        get binary year
C3FD 8D 2C              BSR     BINASC          convert to ascii
C3FF ED C1              STD     0,U++

C401 86 04              LDA     #EOT
C403 A7 C4              STA     0,U             set end of string

C405 35 76              PULS    A,B,X,Y,U restore the important stuff
C407 39                 RTS
                *
                *  ASCBIN - this routine converts 2 ascii chars
                *           to binary
                *
                *      entry: ACC D contains 2 ascii characters
                *
                *      exit:  ACC A contains binary equivalent
                *             Carry is clear if digits are
                *             valid decimal digits (0-9),
                *             otherwise carry is set
                *
                *      accumulators A and B are used and not
                *        restored.
                *
                C408    ASCBIN  EQU     *
C408 81 30              CMPA    #'0             make sure first ascii
C40A 25 1C              BLO     BADIG           char is between
C40C 81 39              CMPA    #'9             0 and 9
C40E 22 18              BHI     BADIG

C410 C1 30              CMPB    #'0             make sure second ascii
C412 25 14              BLO     BADIG           char is between
C414 C1 39              CMPB    #'9             0 and 9
C416 22 10              BHI     BADIG

C418 84 0F              ANDA    #$00001111 keep low 4 bits
C41A C4 0F              ANDB    #$00001111
C41C 34 04              PSHS    B               save 2nd digit
C41E C6 0A              LDB     #10
C420 3D                 MUL                     multiply first by 10
C421 1F 98              TFR     B,A             B -> A
C423 AB E0              ADDA    0,S+            add in 1's digit
C425 1C FE              CLC                     set good RC
C427 39                 RTS                     return
C428 1A 01      BADIG   SEC                     set bad RC
C42A 39                 RTS                     return
                *
                *  BINASC - this routine converts a 1
                *           byte binary number (<= 99 base 10)
                *           to ascii.
                *
                *    entry: ACC A contains binary number
                *    exit:  ACC D contains 2 digit ascii rep
                *
                *    Accumulators A and B are used and not
                *      restored.
                *
                C42B    BINASC  EQU     *
C42B 34 02              PSHS    A               save binary #
C42D C6 08              LDB     #8              # bits to shift out
C42F 4F                 CLRA                    hold BCD value here

                C430    DOBLE   EQU     *
                *
                * Double current BCD result before
                * shifting out a bit from the binary
                * number.
                *
C430 34 02              PSHS    A               double BCD value
C432 AB E0              ADDA    0,S+


C434 19                 DAA                     (in BCD)
C435 68 E4              LSL     0,S             shift out a bit
C437 24 07              BCC     CNK             branch if bit=0
C439 34 02              PSHS    A               add 1 to current
C43B 86 01              LDA     #1              BCD value
C43D AB E0              ADDA    0,S+
C43F 19                 DAA                     (in BCD of course)

C440 5A         CNK     DECB                    done yet?
C441 26 E0              BNE     DOBLE           no, then continue

C443 32 61              LEAS    1,S             clean up stack
                *
                * Convert BCD # in A to ascii
                *
C445 34 02              PSHS    A               save BCD value
C447 44                 LSRA                    cvt 10's dig to ascii
C448 44                 LSRA
C449 44                 LSRA
C44A 44                 LSRA
C44B 8A 30              ORA     #'0
C44D 35 04              PULS    B               cvt 1's dig to ascii
C44F C4 0F              ANDB    #$00001111
C451 CA 30              ORB     #'0
C453 39                 RTS                     return
                        END     START

0 ERROR(S) DETECTED
```

REVIEW OF RMA AND RLINK

A Review of RMA and RLINK
by
Peter Dibble

RMA (Relocating Macro Assembler) and RLINK (Relocating Linker) are new programs from Microware. They are required for C (and probably for future languages from Microware), and are currently bundled with C. Those who already have the C compiler from Microware shouldn't consider purchasing RMA/RLINK — they already have them under the names c.asm and c.link.

Overview

It is easier to explain RLINK's purpose than RMA's. RLINK takes one or more files created by RMA and turns them into an executable module. RMA is a tool which makes writing large programs easier with a moderately good macro facility and a variety of tools which permit a program to be divided into several pieces which can be assembled separately.

This separate assembly is the really important part of RMA. With separate assembly it is easy to build a library of procedures which can be called from any program. Structured programming requires that each procedure be as independent of other procedures as possible. It is much easier to do this when each module has clear connections to other modules — in particular, any shared data should be noted; RMA makes it easy to isolate procedures, and makes it hard to hide shared data.

RMA's Macro Facility

RMA includes the usual conditional assembly statements:

FAIL — Generates an assembler error and a message.
IF/ELSE/ENDC — Do just what they should. ELSE is optional.
REPT/ENDR — repeats a set of statements a specified number of times.

These statements can be used in the body of a program, or in macros. Macros amount to procedures, or specially defined instructions which can be used very much as if they were 6809 instructions. A macro is defined by the MACRO/ENDM statements. A macro can be given parameters which are referred to within the macro by a backslash followed by a number: \1 would be the first parameter, \2 the second, etc. The number of parameters given is available through the special operator \#, and the length of any parameter is available through the operator \Ln where n is the number of the argument whose length is in question.

When a macro needs unique labels, RMA offers the @ operator. This operator returns an @ followed by a number unique to each invocation of each macro.

Here is a sample RMA macro:

```
Swap MACRO      exchanges bytes in memory
*    arg1 — points to memory location
*    arg2 — another location
*    arg3 — the number of bytes to swap (a constant)
     IFNE \#-3 Check the number of args.
     FAIL Swap: must have exactly three arguments
     ENDC
     pshs A,B,X,Y
     leas -1,S Make work space on stack
     leax \1,U address of first variable
     leay \2,U address of second variable
     ldb #\3   number of bytes to swap
     ble @Lx   if none; stop
@lp  lda B,X
```

```
    sta ,S
    lda B,Y
    sta B,X
    lda ,S
    sta B,Y
    decb
    bne @Lp
@Lx leas 1,S  clear work space
    puls A,B,X,Y
    ENDM
```

This macro could be invoked with the statement:

```
Swap Var1,Var2,20
```

which could be used as many times as necessary in a program with Swap defined.

The Separate Assembly Facility

RMA includes statements which define three different "program sections."

The PSECT section contains program code and constants. RMA can only deal with one PSECT per assembly The PSECT statement includes all the data given in the MOD statement in ASM except the module length, but only the entrypoint argument to PSECT is an address. The parameters are:

- name -- Up to 20 byte name for the PSECT
- typelang -- the type/language for the PSECT
- attrrev -- the attribute (ReEnt ?) and revision level of the PSECT
- edition -- the edition number to be used for the executable module.
- stacksize -- The estimated size of the stack for this procedure.
- Entry -- The Label used for the first instruction to be executed in the PSECT.

If the PSECT is the mainline segment of the program being written, all the arguments must have values; for example:

```
PSECT Example,PrgrmMObject,ReEnt+1,1,250,EntryPt
```

Procedures which are used as subroutines must have zeros in some fields; for example:

```
PSECT SubProc,0,0,0,100,0
```

The PSECT section contains only constant data: instruction mnemonics, OS9, fcc, fdb, fcs, fcb, rzb (reserve zero-value bytes, VSECT, ENDSECT, and END. In particular rmb is not allowed in a PSECT.

The VSECT section reserves memory locations. It has two forms:

```
    VSECT DP
```

reserves space in the direct page, and just

```
    VSECT
```

reserves space outside the direct page. The VSECTs are used for the variables that would normally be addressed off the U register in an OS-9 program. Normally only the rmb instruction is used in a VSECT, but for elaborate programs it is possible to have variables automatically initiallized. If you are willing to include the initiallization code in your program (it is included with RMA) you can use fcc, fdb, fcs, fcb, and rzb in a VSECT along with rmb. It is important that there is no official way to know where variables allocated in a VSECT will be relative to other variables. Your program will be able to find its variables, but finding relationships between the addresses of variables at assembly time is hard.

As many VSECTs as convenient can appear in a PSECT.

If VSECT is used inside the PSECT, as it usually is, it will cause the linker to allocate space for the variables in it. If a VSECT is placed outside the PSECT it will make the variables in the VSECT known in the code, but not allocate any storage. This is a useful trick for cases when you know that a block of variables has already been allocated and you want access to all of them. I haven't tried this, and I can't find it in the manual, but Microware declares it will work.

A CSECT is just a way to assign values to names. They are used extensively in the OPTS files for RMA. Only the rmb statement can be used in a CSECT. If the CSECT statement is given an argument, that argument is the starting value in the CSECT, otherwise the values in the CSECT start at zero.

Every program sector must be terminated with an ENDSECT. A PSECT can contain other sectors, but in general sectors should not be nested.

A label can be made globally available by following it with a colon ":" when it is defined. If a label isn't global, it is only known in the PSECT where it is defined. If a label isn't global, it can be used to represent a different thing in each, separately assembled, file.

Speaking of labels: RMA permits labels up to nine characters long and always distinguishes upper and lower case letters.

The files that are produced by RMA, called relocatable files, can be decoded by a program called RDUMP which is included with RMA. RDUMP can give anything from a quick summary to an exhaustive dump of information about symbols referenced and defined in the file being investigated.

Some Internals

Since RMA has no way of telling what offsets RLINK will assign to variables defined in VSECTS, it is often unable to use the small-offset forms of the indexed instructions. References to data in VSECTS are assembled as 16 bit offsets. RMA records information about variables defined and used in a PSECT which is used by RLINK. RLINK goes through the files it is linking filling in the blanks left by RMA.

RLINK accepts a list of files to link and libraries to use. It will link all the files on the command line even if the mainline PSECT doesn't reference anything in them. If there are any references left unresolved, RLINK will search the library(s) for the PSECTs needed to resolve the references. A library is simply a group of PSECTs merged together: the MERGE command does this nicely. PSECTS in a library can call one another, but, since the library is read sequentially, unresolved references must be to PSECT further along in the file, or in another library which will be searched later.

Limitations

I haven't been able to discover an easy way to have RMA calculate the length of a group of variables in a VSECT. The concept of a useful data position counter ("." in ASM) doesn't exist in RMA. There are several counters (Direct Page, Uninitialized data, and initiallized data), and, in any case, the linker has the last word on addresses. I got used to this problem, and I can't think of any way for Microware to design it out of RMA without introducing other problems, but it is a serious problem. The lack of a "." caused other habits I have to generate errors as well.

RMA's inability to determine offsets in a VSECT causes the 16 bit offset instructions to be used more than they are in programs assembled with ASM. These instructions are relatively long and slow. At first this really upset me, but my experience and Microware's indicates that it isn't a significant problem. I converted several very large (5000 to 10000 lines of code) programs from ASM to RMA and they generally got a little smaller. Microware declares that they have converted Basic09 from ASM to RMA, and that it got a little smaller through the conversion. I attribute the small decrease in size to better coding habits that RMA encourages. Still, in the last analysis, programs assembled by ASM can be made to run faster than programs assembled by RMA.

This is really nit-picking, but the command line option which should set the width of the listing which RMA can produce doesn't work. It's not that important, but little problems like that could give a less forgiving person than me a bad impression that would spoil the excellent job done on the really important parts of the product.

I found several problems in the first copy of RMA that I got, some of them quite serious. I now have edition five. If you have an earlier edition, I would strongly recommend getting an update. If you mean to use c.asm as a stand-alone assembler, you should also see to it that you have an up-to-date revision. The problems were tricky things that wouldn't generally show up with correct code, but I haven't been able to uncover any bugs other than the problem with the width of the listing in the current revision of RMA.

Converting programs from the standard assembler to RMA is not as simple as one might think. To start with the standard DEFS files won't work, and Microware didn't include complete DEFS files with RMA. I frequently use "." ... that had to be dealt with. RMA can't handle as many symbols as the standard assembler before the symbol table overflows. This meant that I couldn't just convert a program into RMA, I had to use RMA. A large program MUST be broken down into several PSECTs and assembled in pieces then linked.

Summary

I think RMA/RLINK is wonderful. I am a serious assembly language programmer. I write large programs that take a long time to assemble, and have quantities of chunks of code that I "USE" in assembler programs to prevent myself from having to rewrite commonly used procedures. RMA lets me build libraries, and assemble only the small part of a program that I change. I also care about structured programming, and RMA lets me use that discipline for assembly language programs.

Assembly language procedures to be called from C must be written in RMA, and I have been able to call C procedures from RMA programs. RMA comes with the C compiler, but the documentation that is included in the C manual isn't sufficient to make full use of c.asm/c.link. The information I have given in this review may supplement the C manual enough, but, if not, I would recommend purchasing a copy of the RMA/RLINK manual from Microware.

The standard assembler is easier to use for short and simple programs. RMA has a lot more power, and is correspondingly harder to use. Nevertheless, if you are serious about assembler, RMA/RLINK is important to have. Even if the added structure doesn't mean anything to you, the large amounts of time that you won't spend waiting for big programs to assemble will be worth the investment in money and time that RMA requires.

MIKMAN TO ENTER MIKBUG

-use MIKMAN to enter MIKBUG formatted tapes manually..

G. A. R. Trollope
466, Caswallen Drive,
West Chester, PA 19380

One of my more ambitious projects has been to get the Hemenway Associates disk operating system CP/68 and their structured BASIC - STRUBAL+ up from their Paperbytes software books. While this seems to be a formidable task, the effort is really nothing compared to the effort to develop the software itself. Now, there are two possible ways of entering the programs, from the assembly listings or from the Motorola S1..S9 Format hexadecimal dumps. I think the latter is better to use, as the check-byte gives a measure of protection against typing errors. This is certainly necessary with the aforementioned software, as the listings were prepared with a dot matrix printer that was pretty flakey (aren't they all?). This was particularly true on the bottom row, so that E's and F's are sometimes very difficult to distinguish...but resolution can be obtained by consulting the Assembly listing. Now the standard load routines are pretty unforgiving of typing errors, so that you have to retype the whole line if you make a mistake, even if you realise it. So this was the idea behind MIKMAN, a program designed to make it easier to enter MIKbug style tapes MANually, see listing 1. If the logic of the program is not clear from the listing, it should become clearer as you read the following.

As you enter the data, MIKMAN places it in a buffer. If you realise that you made a mistake, you can backspace up to the error and retype it. You space forward (use DELete) over the correct characters, which are echoed, to resume. Likewise, if you make an error which MIKMAN spots from the checksum, it starts you off at the beginning of the line, and you space forward to the error. After correcting the errors, a carriage return will get MIKMAN to try again. Now if you type a non-hex character, MIKMAN rings the bell and does an automatic backspace for you to retype the character. MIKMAN will ignore a line feed so you can type the corrected input on a new line.

One other major thing MIKMAN does for you is to enter the S1, the byte count, and the loading address automatically for you - it prints these eight characters for you at the beginning of the line. It does this by assuming the byte count for the new frame is the same as the

previous frame, and the loading address follows the last byte of the previous frame. If the byte count is wrong, enter S1, the byte count, and carriage return. If the address is wrong, enter all eight characters. In either case MIKMAN echoes the data at the beginning of a new line, so you can keep the input lined up, which may help you spot errors.

One last thing MIKMAN does to help is to automatically space after you have entered four characters, so the input data forms into columns. To use this effectively, I suggest you very carefully rule lines every four characters in the listing. Use a hard pencil, at least a 4, and sharpen it to a very fine point. Key in the characters in groups of four, staying tuned for the acknowledging space.

Alright, that's all there is to it. Hope listing 2, the S1 dump of MIKMAN, is the last one you'll have to enter painfully. I find I can average somewhat better than one kilobyte/hour with MIKMAN. That's better than 2 baud!

```
PAGE  001   MIKMAN


00010                       NAM   MIKMAN
00020                       OPT   O,NOG
00030                 *******MIKMAN*******
00040                 *ROUTINE TO ENTER MIKBUG FORMATTED TAPES MANUALLY
00050 4D00                  ORG   $4D00
00060                 *INITIALIZE
00070 4D00 7F 4D08  START   CLR   FLAT
00080 4D03 7F 4D0C          CLR   READ+1
00090 4D06 7F 4D10          CLR   BYTECT
00100 4D09 20 51            BRA   LOOP
00110                 *
00115                 *DATA STORAGE
00120 4D0B 0002     NEXT    RMB   2
00130 4D0D 0001     CKSM    RMB   1
00140 4D0E 0002     BUFF    RMB   2
00150 4D10 0001     BYTECT  RMB   1
00160 4D11 0001     FLAG    RMB   1
00170 4D12 0001     FLAT    RMB   1
00180 4D13 0002     READ    RMB   2
00190                 *
00200       E07A    PDATA1  EQU   $E07A
00210       E04A    OUTCH   EQU   $E04A
00220       E1AC    INCH    EQU   $E1AC
00230       E0CC    OUTS    EQU   $E0CC
00240       E0BF    OUT2H   EQU   $E0BF
00250       E1D1    OUTEEE  EQU   $E1D1
00260       A040    STACK   EQU   $A040
00270       E0A3    CONTRL  EQU   $E0A3
00280                 *
00290                 *INHEX CALLING SEQUENCE:
00300                 *   BSR INHEX
00310                 *   BRA ERROR
00320                 *OK.....
00330 4D16 8D E1AC  INHEX   BSR   INCH
00335                 *GET LINEFEED FOR DIRECT RETURN
00340 4D18 10               TBX
00345 4D19 E1 00            CMP   X
00350 4D1C 31               INS
00360 4D1D 31               INS
00370 4D1E 36               PSH A
00380 4D1F 80 30            SUB A  #$30
00390 4D21 2B 10            BMI    CA
00400 4D23 81 09            CMP A  #$9
00410 4D25 2F 0A            BLE    CB
00420 4D27 81 11            CMP A  #$11
00430 4D29 2B 08            BMI    CA
00440 4D2B 81 16            CMP A  #$16
00450 4D2D 2E 04            BGT    CA
00460 4D2F 80 07            SUB A  #7
00462                 *NORMAL EXIT
00464                 *  NOTE: A HAS VALUE, CHAR IS ON STACK
00470 4D31 6E 02    CB      JMP    2,X
00475                 *ERROR EXIT
00477                 * RESTORE STAK
00480 4D33 32       CA      PUL A
00490 4D34 6E 00            JMP    X
00500                 *
00510 4D36 CE 4EF3  NEWLIN  LDX    #MA
                      * PRINT CR
00515
00520 4D39 8D E07E          JSR    PDATA1
00530 4D3C B6 4D10          LDA A  BYTECT
00540 4D3F 36               PSH A
00550 4D40 30               TSX
00555                 *PRINT BYTE COUNT
00560 4D41 8D E0B7          JSR    OUT2H
00570 4D44 31               INS
00580 4D45 CE 4D0B          LDX    #NEXT
00585                 *PRINT ADDRESS
00590 4D48 7E E0C6          JMP    OUT4HS
00600                 *
00605                 *BYTE READS 2 HEX CHARS TO FORM 1 BYTE
00610 4D4B 8D C9    BYTE    BSR    INHEX
00620 4D4D 20 0D            BRA    LOOP
00625 4D4F 31               INS
00630 4D50 48               ASL A
00640 4D51 48               ASL A
00650 4D52 48               ASL A
00660 4D53 48               ASL A
00670 4D54 16               TAB
00680 4D55 8D BF            BSR    INHEX
00690 4D57 20 03            BRA    LOOP
00695 4D59 31               INS
00700 4D5A 1B               ABA
00710 4D5B 39               RTS
00720                 *
00730                 *
00735                 *****MAIN PROGRAM LOOP*****
00740 4D5C 6E A040  LOOP    JMP    STACK
00750 4D5F 8D B5            BSR    NEWLIN
00755                 *
00756                 *SET UP TO READ A LINE OF DATA
00760 4D61 CE 4E75          LDX    #BUF
00770 4D64 FF 4D0E          STX    BUFF
00780 4D67 C6 04    AB      LDA B  #4
00790 4D69 8D AB    AA      BSR    INHEX
00795                 *EXIT FROM LOOP IF NON-HEX CHAR FOUND
00800 4D6B 20 12            BRA    NOTHEX
00805 4D6D 32               PUL A               GET CHAR
00810 4D6E FE 4D0E          LDX    BUFF
00820 4D71 A7 00            STA A  X
00830 4D73 08               INX
00840 4D74 FF 4D0E          STX    BUFF
00850 4D77 5A               DEC B
00860 4D78 26 EF            BNE    AA
00865                 *SEND OUT A SPACE EVERY 4 CHARS
00870 4D7A 8D E0CC          JSR    OUTS
00880 4D7D 20 AB            BRA    AB
00890                 *
00891                 *NON-HEX CHARACTERS:
00892                 *   CR  END OF RECORD
00893                 *   /   DELETE PREVIOUS CHAR
00894                 *   LF  JUST LINEFEED
00895                 *   DEL ECHO THE CHAR USED LAST TIME IN THIS SPOT
00896                 * CNTL/X CANCEL THIS LINE
00897                 *   $9  RETURN TO ASKBUG
00898                 *   $1  CHANGE BYTE COUNT & ADDRESS
00899                 * ALL OTHERS: DELETED, WITH WARNING BELL
00900 4D7F 81 0D    NOTHEX  CMP A  #$D      CR
00910 4D81 27 79            BEQ    INSERT
00920 4D83 81 2F            CMP A  #'/      /
00930 4D85 26 20            BNE    NA
00935                 * / TO BACKSPACE
00940 4D87 FE 4D0E          LDX    BUFF
00950 4D8A 8C 4E75          CPX    #BUF
00960 4D8D 27 CD            BEQ    LOOP
00970 4D8F 09               DEX
00980 4D90 FF 4D0E          STX    BUFF
00990                 *BACK UP PAST SPACE IF BACKSLANT
01010 4D93 5C               INC B
01020 4D94 C1 04            CMP B  #4
01030 4D96 2F 07            BLE    AD
01040 4D98 86 08            LDA A  #8
01050 4D9A BD E1D1          JSR    OUTALL
01060 4D9D C6 01            LDA B  #1
01062 4D9F CE 4AFD  AD      LDX    #MB
01064 4DA2 BD E07A          JSR    PDATA1
01070 4DA5 20 C2            BRA    AA
01080 4DA7 81 0A    NA      CMP A  #$A      LF
01090 4DA9 27 AE            BEQ    AA
01100 4DAB 81 7F            CMP A  #$7F     DEL
01110 4DAD 26 0A            BNE    NB
01120 4DAF FE 4D0E          LDX    BUFF
01130 4DB2 A6 00            LDA A  X
01140 4DB4 BD E1D1          JSR    OUTEEE
01150 4DB7 20 BA            BRA    AC
01160                 *
01170 4DB9 81 18    NB      CMP A  #$18     CAN CNTL/X
01180 4DBB 27 9F            BEQ    LOOP
01190 4DBD 81 53            CMP A  #'S
01200 4DBF 26 2B            BNE    NC
01210 4DC1 FE 4D0E          LDX    BUFF
01220 4DC4 8C 4E75          CPX    #BUF
01230 4DC7 26 23            BNE    NC
01240 4DC9 BD E1AC          JSR    INCH
01250 4DCC 81 39            CMP A  #'9      9?
01260 4DCE 26 03            BNE    NF
01270 4DD0 7E A0A3          JMP    CONTRL
01280 4DD3 81 31    NF      CMP A  #'1      1?
01290 4DD5 26 85            BNE    LOOP
01300 4DD7 BD 4D4B          JSR    BYTE     BITE COUNT
01310 4DDA B7 4D10          STA A  BYTECT
01320 4DDD BD 4D4B          JSR    BYTE
01330 4DE0 B7 4D0B          STA A  NEXT     ADDR HI
01340 4DE3 BD 4D4B          JSR    BYTE
01350 4DE6 B7 4D0C          STA A  NEXT+1   ADDR LO
01360 4DE9 7E 4D5C  NX      JMP    LOOP
01370                 *
01375                 * ADDRESS NON-HEX CHAR
01380 4DEC CE 4F02  NC      LDX    #MC
01390 4DEF 81 20            CMP A  #$20
01400 4DF1 2C 03            BGE    ND
01410 4DF3 CE 4F07          LDX    #MD
01420 4DF6 BD E07E  ND      JSR    PDATA1
01430 4DF9 7E 4D69          JMP    AA
01440                 *
01450                 *
01455                 *TRANSFER BUFFER TO MEMORY
01460 4DFC CE 4E75  INSERT  LDX    #BUF
01470 4DFF FF 4D14          STX    NEXTA
01480 4E02 B6 4D10          LDA A  BYTECT
01490 4E05 16               TAB
```

```
01500 4206 FB 4D0B        ADD B  NEXT
01510 4209 FB 4D0C        ADD B  NEXT+1
01520 420C F7 4D0D        STA B  CKSM
01530 420F 80 06          SUB A  #2
01540 4211 B7 4D21        STA A  BYTEC
01550 4214 FE 4D0A        LDX    NEXT
01560 4217 6D 32    1A    TST    BYTE,X
01570 4219 7A 4D11        DEC    BYTEC
01580 421C 27 09          BEQ    ID
01590 421E A7 00          STA A  X
01600 4220 A1 00          CMP A  X
01610 4222 26 15          BNE    ID
01620 4224 08             INX
01630 4225 20 F0          BRA    1A
01640 4227 7C 4D0B  2B    INC    CKSM
01650 422A 26 05          BNE    1C
01660 422C FF 4D0B        STX    NEXT
01670 422F 20 20          BRA    6A
01673                *
01676                * CHECKSUM ERROR
01680 4231 CE 427A  1C    LDX    #ML
01690 4234 BD A07A        JSR    PDATA1
01700 4237 20 B0          BRA    RE
01703                *
01706                * MEMORY ERROR
01710 4239 CE 4213  ID    LDX    #AF
01720 423C BD A07A        JSR    PDATA1
01730 423F FF 4D14        STX    HEXTA
01740 4242 CE 4D14        LDX    #HEXTA
01750 4245 BD B6C0        JSR    OUT4HS
01760 4248 7A A0E3        JMP    CONTRL
01770                *
01780                *
01785                * FORM 1 BYTE FROM 2 HEX CHARS IN BUFFER
01790 424B FF 4D12  BYTEH STX    SAVX

01800 424E FE 4D14        LDX    HEXTA
01810 4251 A6 16          LDA A  IHexAA
01820 4253 48             ASL A
01830 4254 48             ASL A
01840 4255 48             ASL A
01850 4256 48             ASL A
01860 4257 16             TAB
01870 4258 BD 0F          BSR    IHexAA
01880 425A 1B             ABA
01890 425B 16             TAB
01900 425C FB 4D0D        ADD B  CKSM
01910 425F F7 4D0D        STA B  CKSM
01920 4262 FF 4D14        STX    HEXTA
01930 4265 FE 4D12        LDX    SAVX
01940 4268 39             RTS
01950                *
01955                * GET A HEX CHAR FROM BUFFER, &CONVERT
01960 4269 A6 00    IHexAA LDA A  X
01970 426B 08             INX
01980 426C 80 30          SUB A  #$30
01990 426E 81 09          CMP A  #$9
02000 4270 2F 02          BLE    IG
02010 4272 80 07          SUB A  #7
02020 4274 39       IG    RTS
02030                *
02040                **INPUT BUFFER
02050 4275 0080     BUF   RMB    126
02060                *
02070                *
02080                *LITERAL POOL
02090 42F5 0D       M-    FCB    $D,$A,0,0,0,'$,'1,4
02100 42FD 08       M>    FCB    8,8,7,8,4
02110 4706 07       MC    FCB    7,8,7,8,4
02120 4707 07       MD    FCB    7,7,4
02130 470A 43       ML    FCC    'CKSM ERR.
02140 4712 04             FCB    4
02150 4713 07       MF    FCB    7,7,7,7,8D,4
02160 4719 4D             FCC    'MEMORY ERROR AT $.
02170 472A 04             FCB    4
02180                     END

TOTAL ERRORS 00000
```

```
S1132D007F2D0B7F2D0C7F2D1020514202D14D0B1
S1132D10444154413 1E0BDE1AC30EE003131368004
S1132D20302B1081092F0A81112B0881162E048063
S1132D30076E02326E00CE2E75BDE07EB62D1036C3
S1132D4030BDB0EF31CE2D0B7EE0C88DC9200D31E2
S1132D5048484848166DBF2003111B398EA0408D4A
S1132D60D5CE2EAFFF2D0EC6048DAB201232FE2D14
S1132D700EA70006FF2D0E5A26EFBDE0CC20E881F7
S1132D800D2779810826207FE2D0E8C2EAF27CD0924
S1132D90FF2D0E5CC1042F078608BDE1D1C601CE0C
S1132DA02E7DBDE07E20C2810A27BE817F260AFED9
S1132DB02D0EA600BDE1D120BA811827F8153268C
S1132DC02BFE2D0E8C2EAF2623BDE1AC81392603BC
S1132DD07EE0E381312685BD2D4BB72D10BD2D4BF3
S1132DE0B72D08BD2D4BB72D0C7E2D5CCE2E8681C1
S1132DF0202C03CE2E8BBDE07E7E2D69CE2EAFFF20
S1132E002D14B62D1016FB2D0BFB2D0CF72D0D805C
S1132E1002B7E0AA9A2A0D8D327A2D112709A70033
S1132E20A10026150820F0C2D0D26C5FF2D0B2072
S1132E30E0CE2E8EBDE07E20B0CE2E97BDA07EFFB4
S1132E402D14CE2D14BDE0C87EE0B3FF2D12FE2D1F
S1132E5014BD164046464648168D0F1B16FB2D0DF786
S1132E602D0DFF2D14FE2D1239A6000880306109B6
S1132E702F0280073900D0A0000005331045F0804153
S1132E800000000000000070B2F0804070704434B54
S1132E90534D20455252040707070707D0A4D454D6F
S1122EA04F7259204552524F52204154202404 7E
```

Listing 2 MIKMAN object code (S1..S9)

BIT BUCKET

MOTOROLA 68000 SOFTWARE SUPPORT

PRESS INFORMATION

EDITORIAL CONTACT: DAVID ANDERSON
(602) 994-6881

READER CONTACT: BILL LOWERY
(602) 829-3507

MOTOROLA ANNOUNCES DELIVERY TO AT&T OF SYSTEM V/68™, THE FIRST JOINTLY DEVELOPED UNIX™ OPERATING SYSTEM

Washington, D.C....January 18, 1984...Motorola announced that its arrangement with AT&T to implement UNIX System V for Motorola's M68000 family of high performance microprocessors has been, for all intents and purposes, completed. Motorola has delivered the product to Bell Labs for final acceptance testing. Because the two companies have enjoyed continuous communication throughout product implementation, no problems are anticipated during the acceptance process.

As Motorola and AT&T had previously agreed, the operating system is a faithful, functional equivalent to the UNIX System V product developed by Bell Labs and released by AT&T for Digital Equipment Corporation minicomputers. AT&T is expected to market the product under the name "UNIX System V, M68000 Version Operating System." Motorola will market a derived version, called SYSTEM V/68 Operating System. The SYSTEM V/68 Operating System is the host environment for Motorola's EXORmacs™ and VME/10™ development systems, and will be available to manufacturers of M68000 based microcomputer systems immediately after product acceptance by AT&T. Source code for the SYSTEM V/68 Operating System will be available from Motorola to holders of UNIX System V, M68000 Version source licenses, which may be obtained from AT&T Software Marketing and Sales. Object redistribution licenses for the SYSTEM V/68 Operating System will be available from Motorola. As agreed upon with AT&T, Motorola will provide product support for both UNIX System V, M68000 Version, and SYSTEM V/68 operating systems at the source code level.

In addition, Motorola will provide support for SYSTEM V/68 object code operating on Motorola development systems. Domestic support will be performed by Motorola's wholly-owned subsidiary, Four Phase Systems, which has extensive field engineering service locations throughout the U.S. International support will be coordinated by Motorola's subsidiaries in Munich (Motorola GmbH) and Tokyo (Nippon Motorola Ltd.).

The delivery to AT&T of the completed UNIX System V derivation for the M68000 family illustrates Motorola's allocation of internal engineering, marketing, and product support resources. Said Tom Beaver, Vice President and Director of Motorola Microsystems, "Motorola realized long ago that the UNIX Operating Systems and the M68000 family of microprocessors were a natural match." The MC68000 is a proven, very high performance microprocessor, currently available in speeds up to 12.5 MHz. The MC68010 microprocessor combines the speed and instruction set of the MC68000 with extensions to support virtual memory systems. "The fact," said Beaver, "that Motorola is the first semiconductor

manufacturer to deliver a jointly developed UNIX System V derived product to AT&T demonstrates our commitment to satisfy and support the M68000 marketplace with the UNIX Operating System."

"In addition to the SYSTEM V/68 Operating System Motorola will continue to support and enhance our current M68000 software offerings, which include VERSAdos™ and RMS68K™. To provide a wide range of software development tools for M68000 family users, Motorola will also enter into arrangements with other software suppliers to design and market languages, operating systems, and utilities that customers require."

The SYSTEM V/68 Operating System now supports the MC68000 and MC68010 microprocessors. An enhanced version, supporting the MC68020 32 bit processor, will be introduced when the MC68020 becomes available. As the M68000 family evolves, the SYSTEM V/68 Operating System will continue to provide a complementary high performance operating system environment.

ADVANCE INFORMATION
MICROSYSTEMS

MOTOROLA

SYSTEM V/68 Operating System Software

- The Standard UNIX-derived Operating System for the M68000 Microprocessor Family
  - — Small, Flexible Kernel with Performance Optimized for M68000 Family.
  - — Command Interpreter ("Shell") Offers Powerful Facilities for Interactive Control
  - — Extensive Set of Programming Languages (C, FORTRAN, BASIC, etc.) and 68000 Assembler
  - — Text Processing Tools
  - — Electronic Mail
  - — Communications and Networking Support
  - — Programmers Workbench
- Motorola Pascal 2.1, Macro Assembler, and Linker/Loader Optionally Available
- Languages Support MC68000, MC68010, and MC68008 Microprocessors
- Support for Motorola Memory Management Devices
- Executes on Motorola Development Systems
- Source Code Available on Motorola Development Systems and Non-Motorola Systems Capable of Reading UNIX cpio Format Media (AT&T UNIX System V Source License Required)

SYSTEM V/68 is derived from UNIX System V, M68000 Version, a jointly developed product of Western Electric Company, Inc. and Motorola Inc.

The SYSTEM V/68 Operating System is the standard UNIX-derived Operating System for the M68000 family of microprocessors. It offers a small compact kernel, which provides process scheduling and I/O facilities to all programs. In addition, a powerful command shell for interactive system controls and an extensive set of utility programs for many tasks, such as program development, text processing, electronic mail, and networking support are included.

M68NNCBSV

Technical Data

Host Systems

The SYSTEM V/68 Operating System is available as the host environment on Motorola development systems. The EXORmacs is a multiuser system capable of supporting up to ten users simultaneously. The VME/10 System is a single-user system. Hard disk is required for SYSTEM V/68. Future Motorola Microsystems development systems will also be supported by the SYSTEM V/68 Operating System.

Languages

As an integral part of SYSTEM V/68, the C language is offered. C language has developed into one of the most popular commercial programming languages, and is used frequently in developing portable application software. SYSTEM V/68 offers significant enhancements to the C language, along with several new language utilities. CXREF, a new cross reference program, and CFLOW, a new flow analysis program, are just two of the new utilities offered. SYSTEM V/68 also includes a FORTRAN 77 compiler as well as an M68000 assembler and linker/loader.

Programmer's Workbench

The Programmer's Workbench utilities support the development of large software systems in a professional manner. They include the Source Code Control System (SCCS), which provides facilities to store, update and retrieve all versions of source code modules.

Text Processing

Text processing utilities include the *ex/vi* full-screen editor, NROFF and TROFF text formatters, a spelling checker, and programs for formatting tables and mathematical equations. The *ex/vi* editor supports a large number of existing terminals, including the Motorola EXORterm 155, through the use of the *termcap* terminal data base. *Termcap* entries for new terminals may be added by the user.

Communications

SYSTEM V/68 utilities provide support for electronic mail, communications, and networking. Electronic mail allows users to communicate with one another, using the system as a mailbox or as a bulletin board. The communications utilities allow a SYSTEM V/68 user to communicate to mainframe computers. Networking support allows several computers to be linked together, either through dedicated links or by dial-up telephone connections. With these utilities, Motorola development system users can communicate with one another. In addition, target systems may be developed with similar capabilities. In SYSTEM V/68, interprocess communications routines have been added. These include shared memory, messages, and semaphores. Also included is an IPC/remove command which removes message queues, semaphores, and shared memory identifiers from the system.

Ordering Information

Part Number	Description
[illegible]	SYSTEM V/68 software supplied on CMD cartridge. SYSTEM V/68 includes the following object code modules — M68000 System V/68 Operating System — M68000 C Language Compiler, Assembler, and Linker — Full set of System V/68 documentation — OEM Configuration Guide — Users Guide Object code will be supplied as bootable load modules and the kernel also as relocatable, partitioned, and unlinked modules so that the OEM can reconfigure the SYSTEM V/68 operating system without purchasing source code. In addition, a sample source code driver is included.
M68NNMBSV	Same as above except on 25 Mbyte (removable) LARK cartridge
M68NNCSSV	SYSTEM V/68 software supplied on CMD cartridge. System V/68 includes the following — M68000 System V/68 Operating System (SOURCE CODE) — M68000 C Language Compiler, Assembler, and Linker — Full set of System V/68 documentation — OEM Configuration Guide — Users Guide
M68NNMSSV	Same as above except on 25 Mbyte LARK cartridge
M68NNQSSV	Same as above except supplied on 1600 BPI magnetic tape in UNIX cpio Format
M68KV7	Four-channel RS-232C Communications Module (full duplex). Provides serial ports for adding up to four additional user terminals for System V/68 on an EXORmacs Development System host. Note: the basic EXORmacs System supports two user terminals.
M68NNCSSVVAM	Combination Package — M68NNCSSV + M68KV7
M68NNCBSVVAM	Combination Package — M68NNCBSV + M68KV7

MOTOROLA Semiconductor Products Inc. [redacted]
P.O. BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

PRESS INFORMATION

MOTOROLA
EDITORIAL CONTACT: DAVID ANDERSON
(602) 994-6881

MOTOROLA
READER CONTACT: FLO HARTELOO
(602) 829-3068

DIGITAL RESEARCH
EDITORIAL CONTACT: ELIZABETH CHANEY
(415) 494-2030

DIGITAL RESEARCH
READER CONTACT: JUDY MERVIS
(408) 646-6218

MOTOROLA UNVEILS CONCURRENT DOS—UNIX— IBM PC/DOS PORTABILITY IN M68000 BASED SYSTEMS

Washington, D.C. January 18, 1984...In a move enhancing standardization of popular operating system software, Motorola has commissioned Digital Research, Inc. (DRI) to implement their Concurrent DOS Operating System and standard languages to Motorola's M68000 family of microprocessors. According to both companies, this will enable full portability of applications software between UNIX™ System V (which is already operating on Motorola's VME/10™ Desktop Microcomputer) and Concurrent DOS on M68000 based systems. In addition, because of Concurrent IBM DOS mode, the application software library developed for these systems in a high-level language will be portable to M68000 systems as well.

The recently signed agreement calls for Motorola and DRI to develop 19 software packages for the M68000 family in CP/M, Concurrent DOS, and UNIX System V. Concurrent DOS, which is written in "C" language, is a multi-tasking operating system which provides PC/DOS support. It includes windowing, LAN supports, graphics, and is designed for single or multi-user microcomputers. Concurrency means the user can accomplish several tasks at the same time with windowing permitting multiple screens to be displayed simultaneously. Seven of the packages to be developed will support the M68000 UNIX System V operating system. The packages being developed include DRI's popular programming languages. These language products provide the application portability from CP/M and concurrent DOS to UNIX System V. The packages, CP/M, Concurrent DOS, and library of languages, to be fully maintained and supported by DRI, will be marketed by both organizations.

According to, Tom Beaver, Director of Motorola Microsystems Operations, "The agreement reinforces Motorola's commitment to support the M68000 microprocessor family with state-of-the-art operating systems that facilitate implementation of the myriad of applications software developed by third party vendors. This latest development along industry-standard lines, and is being combined with development and support of advanced semiconductor components to provide a portable, performance oriented environment for applications software." He added, "This is the first in a series of moves to provide complete portability between UNIX System V, Concurrent DOS, the VME/10 standard, and other major operating system software libraries."

When the project is completed, by the end of the year, it will be possible to port source-code from Concurrent DOS to UNIX System V, or vice versa, with little or no code conversion required. This results in significant time and monetary economies for designers, manufacturers, and users throughout the product development cycle, allows new products to enter the market earlier, and provides for continued use of software products on obsolete systems. These activities herald the increased usefulness of software which, once developed on one system can be easily transported to another.

During the first quarter, a number of CP/M products will be introduced for users who plan immediate design-ins. These include:

* CP/M-68K -- VME/10
Digital Research C
Pascal MT+
CBasic Compiler

Concurrent DOS and UNIX System V/68 products to be available by the end of the year will offer a wider range of languages and utilities for most applications, yet will be source-code compatible with CP/M-68K languages. Additionally, a port to CP/M-68K is a precursor to a port to Concurrent DOS-68K. This means that OEM's and software vendors can start development for M68000-based products in the near term and not suffer any wasted effort.

* Concurrent DOS VME 10
Digital Research C
Pascal MT+
CBasic Compiler
Fortran 77
PL/I (G)
Basic Interpreter
GSX Graphics Software

The same high level languages listed above, supporting Concurrent DOS on the VME/10, will be developed to support UNIX System V/68. These products will also be available by year end.

Digital Research Inc., Pacific Grove, California provides system software and software related products for the full spectrum of microcomputers in business, professional, and consumer markets.

QUESTIONS AND ANSWERS

Q. Why Concurrent DOS and UNIX?

A. As many industry standard operating systems as possible need to be available for the 68000 family to make it easy to use in target systems. The Concurrent DOS and UNIX are both industry standards and with the common language tie they are even more beneficial to users. They can pick the operating system that makes sense and not have to rewrite their application code.

Q. DRI is working with a number of major manufacturers. What does this mean in terms of portability from/to these foreign systems?

A. The purpose of this agreement is to make as many software packages as possible available on a wide variety of hardware. This means that a vast number of software packages which are currently only available on one box - for instance the IBM-PC - can be made available on many boxes. These boxes may be based on different processors. From Motorola's viewpoint, it enhances the desirability of the 68000 if the considerable quantity of software which is available for the IBM-PC is available on 68000-based systems.

Q. Will presently available CP/M packages be portable?

A. Those presently available packages which are written in one of DRI's current high level languages should be portable to Concurrent DOS and UNIX. Those written in assembler would have to be recoded in a high level language.

Q. Since you apply this system to only the VME/10, how does it relate to other 68000-based machines/systems?

A. CP/M and Concurrent DOS should be easily ported to other 68000 based systems by creating an I/O system to support the peripheral devices configured for that system. This would typically be two months of effort for CP/M and four months of effort for Concurrent DOS.

Q. What conversions will early adapters using CP/M face when Concurrent DOS is available?

A. If a software vendor's product is implemented using one of DRI's current compilers for CP/M-68K (i.e. C, Pascal or CBasic), then they will merely have to be recompiled and linked again to run on Concurrent DOS.

MOTOROLA GMBH
GESCHÄFTSBEREICH HALBLEITER

Dear Don,

Here's an item of news for all your readers outside the U.S.A. The European branch of Motorola Semiconductors have designed a Universal Floppy Disk Controller and a Winchester Controller for OS-9 in EXORciser format (Microcodule). We have the boards running in our EXORset 165 and 161 products at present we have been marketing EXORset with OS-9 with great success since July 1983. The advantage of the EXORset is its not-so-high cost (£ 5000 for a unit with 56 KB RAM, 1.2 MB floppy disk, OS-9 Level 1, BASIC09 and many other bells and whistles). However, apart from EXORset usage existing EXORciser and Microcodule systems can also be converted to OS-9 with the new disk controllers. We intend to actively market these boards in Europe, Australia, Japan and S. Africa. If any of your readers in these countries would like some advance data, they can write me at the address below.

For your information, I enclose a copy of Motorola's European marketing bulletin where 68 Micro Journal gets special mention for services to 6809 users everywhere!

Best Regards,

John Simms
Motorola MicroSystems Europe
Muenchnerstrasse 18
8043 Unterfoehring
WEST GERMANY

MOTOROLA

Microsystems Update
No. 21
December 1983

MICROSYSTEMS

Microsystems Update c/o John Simms, Motorola Microsystems, 8043 Unterfohring, Muenchenerstraße 18, West Germany

VERSAdos

VERSAdos 4.2
Pascal 2.3
b8343 FFP 1.1
DAD PP 1.2
MDS-400 S/W 4.10
Tutor Source 1.3
Fortran 2.5
6809 xPascal 1.20
6800-01 xasm 1.1
6805 xasm 1.1
6809 xasm 1.1

OS-9 A' Production Release

SYSTEM V/68 Beta Site Release

OS-9 Now in Paperback

Yes, there is yet another reason to sell EXORset and OS-9. We've anticipated that moment when you told the prospect all about OS-9, you gave him the brochure, and a match-book and you've shown him the slide presentation, and he still won't believe you because it sounds that good it can't be true! For many applications OS-9 is the ideal solution. It's small, modular, re-entrant, with many UNIX-like features. Maybe the operating system has to be studied to be believed, there is nothing else quite like it. One way to convince the customer, give him the User's Guide to read. The paperback version is available now. Place orders for 15 copies min. Part Number is M68OS9UM/P1 and M68BASIC09-P1

Keep informed about OS-9 Third Party Software. Subscribe to 68 Micro Journal 5900 Cassanda Smith PO Box 849 Hixson Tennessee TN 37343 USA. This monthly magazine contains feature articles and reviews of interest to OS-9 users and publishes advertisements for OS-9 software suppliers.

MVME315 AVAILABLE NOW

Don't wait to place your orders on this important new VMEmodule. The MVME315 is an intelligent disk controller that has the capability of attaching up to four floppy disk drives and any SASI bus compatible controller (such as a Winchester) to a VME system. Software support is VERSAdos. The system is in Beta site testing and was demonstrated at the Motorola stand in Paris. The 1-off resale price is $1200.

UNIVERSITY of PENNSYLVANIA
PHILADELPHIA 19104

Department of Chemical Engineering

Towne Building D3
220 S. 33rd Street
Area Code 215-898-8351

January 18, 1984

Leo Taylor
189 Twin Brook Rd.
Hamden, Conn. 06514

Dear Leo:

Your COPY utility program is a real blessing. I have been using it for several weeks to great advantage. However, I have found a minor bug. It can be demonstrated by, for example, the command

P COPY LF 2,0,178-248

(This was an effort to get a partial directory listing of some files on my Winchester.) When in the "file number" mode, COPY does not correctly send carriage returns to the printer, with the result that you wind up with a big black smudge at the end of the one and only line of the printed listing. The correction I made is simple: a JSR PCRLF before the line JSR OUTNAM which is at $0822 in the '68 Micro listing. The extra bytes also mean that another change must be made: BSR CPYFIL must be changed to LBSR CPYFIL in the line $07D7. Finally, to prevent double spacing in printed alphanumeric file lists, JSR PCRLF must be omitted from the line at $0744. Several other commands in that series could also be omitted, but it is not necessary.

If I could change one feature of COPY (and I may eventually) it would be to permit skipping missing file numbers. For instance, if I hadn't seen that I had previously deleted file 200 and used the above command line, COPY as now written would terminate and declare an error. I would prefer that it just skip the missing file number and keep on going within the range declared.

Sincerely,

David

David J. Graves
Associate Professor

cc: Don Williams, Sr., '68 Micro Journal

djg/swtpc

PHILIP C. NUNN, Consultant
Engineering & Research Management

201 Netherfield, N.W.
Comstock Park, Michigan 49321
Telephone (616) 784-5732

6 January 1984

Larry E. Williams, Editor
'68' Micro Journal
P.O. Box 849
Hixson, TN. 37343

REACHING FOR PROGRAM TRANSPORTABILITY

Recently, I received a letter from one of your readers who asked my advice on moving some programs from an IBM-PC to a SS-50 bus computer. Since I am asked the question about how to transport programs more frequently as time goes by, I think that my response is of general interest to your readers. The body of my response is in the following paragraphs:

In general, my answer to your question about running CPM-86 programs on a 6809, 50-bus computer is that there is a very low probabiltiy of success, on the order of about 5%. The reason for this low probability is the extreme difference between the 8086 processor's architecture and that of the 6809. John Wakerly, in his excellent book, *Microcomputer Architecture and Programming* (John Wiley & Sons, 1981), provides a lucid explanation of this difference. In summary, they don't even work the same way.

The only possible way to make such a transfer is via a tightly limited process:

1. You must have the source program for your 8086 in BASIC, FORTRAN, PASCAL or some other high level language. Programs written in assembly language must be completely rewritten for the 6809.

2. Transfer these source programs from your IBM-PC to your SS-50 bus computer via a telephone and modem hook-up. This circumvents the horrible problems of disk format incompatibilities.

3. You must now have a comparable compiler or interpreter for the same high level source language on your SS-50 computer, so the programs can be recompiled for it.

4. Before recompiling on the SS-50 bus computer, edit the source program to change any syntax and logic differences between the two language compilers. This is almost always necessary in those parts of a program which work with disk files.

5. Recompile the edited program on the SS-50 bus computer. The inevitable error messages will tell you of the syntax changes which you missed. Continue to correct these until an errorless compile is achieved.

6. Run the compiled program on your SS-50 bus computer. At this point, you still have only a 50% chance that the program will run to completion and do what you want it to. You probably will still have some logic problems from the translation.

Above all, expect this to be a frustrating process. You are working against a long-standing tradition of computer hardware and software designers. Transportability benefits only we computer users.

The difficulties of transportability can be minimized by writing programs in standardized high level languages like FORTRAN and PASCAL. This is a practice which I have adopted because I don't know what brand computer or operating system I'll have to work with next.

In the case of transporting programs from an IBM-PC or other CP/M compatible computer, installing a Z80 processor board, like the Meta Lab Z809, on your SS-50 bus may reduce the anxiety of Step 4. The 8080, Z80, 8088, 8086 family of processors have a similar architecture. Also, an effort has been made by some of the major CP/M software vendors, such as Digital Research, Microsoft and MicroPro,

HELIX

64K 6809 Computer	$2395.00	64K 68008 Computer	$2495.00
256K 6809 Computer	2895.00	256K 68008 Computer	2995.00

Other systems available

20 Megabyte 5" add on Winchester System	$2595.00
64K CMOS Static memory board with battery backup	395.00
DMA 5" and 8" Floppy Controller with built in Winchester controller I/O	695.00
DMA 5" and 8" Floppy Controller $495.00 6809 CPU Board	495.00
68008 board for SS-50 595.00 CP/M-68K	350.00

Need FLEX, UniFLEX, OS-9 Level I, or OS-9 Level II? We have a system for you!

ELEKTRA COMPUTER CABINET THE LARGEST SS-50 COMPUTER CABINET AVAILABLE! Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 6-3/4" high. Heavy duty A.C. line cord, A.C. fuse holder, EMI filter, Fan with filter. Back panel has 10 cutouts for D type data connectors. Front panel has key on/off power switch, 2 illuminated push button switches (Reset and NMI/Abort), and two cutouts for 5-1/4" disk drives. $250.00

Filler Plate for 5-1/4" drive opening $10.00 Fan Filter $10.00

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 3a @ 16v, 3a @ -16v. 3 primary inputs for light, rated, and heavy loading.
220v Version $200.00 110v Version $175.00

DISK REGULATOR BOARD WITH CABLES Side-by-side version for 2 floppy drives $50.00
Heavy duty version for 1 Winchester drive and 1 floppy drive. $75.00

AUXILIARY POWER SUPPLY to power second Winchester drive $125.00

ENGINEER'S "FUN BOX" BY ELEKTRA Computer cabinet with high quality 10 amp power supply, EMI filter, fan. Large enough to hold the standard size SS-50 type motherboard. $225.00

ELEKTRA UNIVERSAL 68-30/68-50C MOTHERBOARD Heavyweight 0.125" thick, 18" long by 9" wide. 11 memory (50 pin) slots, 8 I/O (30 pin) slots. Complete address decoding and selection, as well as extended address capability, for I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. 1" spacing between all memory and I/O slots. On board baud rate generator with low and high range, providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines. Slow device circuitry permitting 1 MHz 30 pin disk controllers to run with 2MHz 50 pin CPU boards.

Mounting hardware	$5.00	Bareboard w/documentation	$80.00
Kit w/gold connectors	$320.00	Assembled w/gold connectors	$380.00
Kit w/tin connectors	$2.00	Assembled w/tin connectors	$300.00

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, motherboard with gold square pin connectors, assembled and tested. $850.00

ELEKTRA CPU 09 Use either the 6802 or 6808 (to run 6800 software) or 6809. Has provision for up to 3 2715 Eproms, 1K scratchpad, MC6840 triple timer, and an optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges. Versions of OS-9 level 1 are available.

Bareboard	$50.00	Kit $225.00	Assembled $275.00

Optional Baud Rate Generator $25.00

ELEKTRA DPB DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per port. RTS, CTS, DTR, DCD, IRQ, FIRQ/NMI, and baud rate can be appropriately implemented for each port.

Bareboard	$20.00	Kit $60.00	Assembled $80.00
Cable with mounting hardware (two needed per board)			Each $25.00
Cable	Each 20.00	Mounting hardware per cable	$5.00

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request line for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line.

Bareboard	$20.00	Kit $60.00	Assembled $80.00
Cable with mounting hardware (two needed per board)			Each $25.00
Cable	Each 20.00	Mounting hardware per cable	$5.00

ELEKTRA 64K STATIC RAM/ROM MEMORY BOARDS with gold connectors (tin available) Assembled and tested. With 56K RAM $269.00 With 64K RAM $299.00

ELEKTRA UNIVERSAL SUPER FLOPPY CONTROLLER THE BEST 30 PIN FLOPPY DISK CONTROLLER THAT YOU CAN BUY! Controls up to four 5-1/4" drives and four 8" drives for a total of eight system drives. (FLEX system limit is four drives.) Single density or double density, 1MHz or 2MHz, 6800 or 6809. (Double density 8" must be at 2MHz; all other combinations of performance are possible.) Analog phase locked loop data separators with separate adjustments for 5" and 8" drives. Analog write precompensation circuit with separate adjustments for 5" and 8" drives. Designed to meet the data hold requirements of Western Digital floppy controller IC. Assembled and tested. $275.00

Disk with drivers and formatting utilities (Specify 6800/9, FLEX/OS-9) 30.00

ELEKTRA WINCHESTER SYSTEMS THE BEST WINCHESTER SYSTEMS THAT YOU CAN BUY! Has automatic error detection and CORRECTION of up to 11 bit burst errors. SS-50 bus, extended addressing capabilities, DMA, on board sector buffer, drivers included for 6809 FLEX or OS-9. Specify whose version of FLEX that you are using. Drivers for FLEX2 (6800) are available for an additional $100.00. Price includes host interface, controller, drive(s), and cables.

12 Megabyte single drive sys	$2295.00	24 Megabyte dual drive sys	$3595.00
19 Megabyte single drive sys	$2995.00	38 Megabyte dual drive sys	$4695.00

(19 Megabyte drives are the largest that can be supported by FLEX.)

ELEKTRA HO-5 Cabinet for dual 5-1/4" floppy drives with power supply, line cord, fuse, power switch, and power cables to drives. 150.00

ELEKTRA HO-5W As above but with EMI filter, fan, and heavy duty power supply. Powers 1 floppy and 1 Winchester. 199.00

5" ribbon cable for dual outboard 5-1/4" disk drives	40.00
5" ribbon cable for dual inboard 5-1/4" disk drives	35.00
Custom cables available	phone

ELEKTRA HO-8 Dual drive cabinet, EMI filter, fan, power supply, and power supply cables for 8" drives. 350.00
8" ribbon cable for dual 8" disk drives 45.00

ELEKTRA 30 PIN PROTOTYPING BOARD 20.00
ELEKTRA 50 PIN PROTOTYPING BOARD 40.00

GOLD 10 PIN CONNECTORS (Specify male with square pins or female) 1.50
TIN 10 PIN CONNECTORS (Specify male with square pins or female) .50

ELEKTRA is a trademark of AAA Chicago Computer Center.
FLEX and **UniFLEX** are trademarks of Technical Systems Consultants, Inc.
HELIX is a trademark of Hazelwood Computer Systems.
OS-9 and **BASIC09** are trademarks of Microware Systems Corp.

Dealer for ELEKTRA, HELIX, SSB, SWTPC, Microware Systems Corp., and Technical Systems Consultants, Inc.

AAA CHICAGO COMPUTER CENTER
120 CHESTNUT LANE • WHEELING, IL 60090 (312) 459-0450

Technical consultation available 4 PM to 5 PM most weekdays. Closed evenings and weekends.

TERMS Minimum order $20.00. Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 6% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Checks must be drawn on a U.S. bank. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 6 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology. We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices and inventory are subject to change without advance notice.

ELEKTRA™ SOFTWARE (All of our software is copyrighted and all rights are reserved. Source is either supplied or optionally available at extra cost so that the purchaser can modify our programs for his own use. Licensing, however, is required for commercial resale.)

SUPER MODEM PROGRAM Single character commands. No interrupts required. Transmit manually or transmit disk files (text) of any length to distant computer. Receive and save disk files (text) on local disk system. X-on/X-off supported. Tested for full duplex at speeds up to 9600 baud. Half duplex option. Echo option. Replaces CR with CR/LF (user option). Slow disk file transmit option.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8".
Instruction Manual and disk with both sources and object code $75.00

STANDARD MODEM PROGRAM Same as Super Modem program above but without ECHO option, CR/LF for CR option, slow disk file transmit option, nor X-on/X-off option. Specify 6800 or 6809.
Manual with instructions, source listing, and flow chart 10.00

OS-9 Configurable Modem Program (Sorry, source is not available) 100.00

ORDER — WRITE UP COMPUTER PROGRAM Screen oriented write up form with cursor editing, disk save and load, printer command using easily available universal print-out forms. Phone for more details. Available for 6809 FLEX. $100.00

ALL IN ONE
Editor — Text Processor — Mailing Labels — Mailing Lists — Multiple Form Letters. Use any CRT terminal and printer — Best Package For The Money Anywhere!
Specify 6800 or 6809, SSB or FLEX™, 5" or 8" 75.00
Printed source listing is available for an additional 35.00
All-In-One, Write'n spell, and Spell'n Fix package 250.00

Software by Technical Systems Consultants, Inc.	FLEX(TM) Adapt. Guide	Source (List)	Source (Disk)	Man. Only	Object w/Man.	UniFLEX(TM) Add. Man.	Man. Only	Object w/Man.
Gen DOS w/Edit & ASMB	25	—	—	50	250	—	—	—
SWTPC DOS w/Edit & ASMB		—	—	25	150	40	100	550
Advanced Programmers Guide		—	—	25	—	—	—	—
Editor		100	250	25	50	—	—	—
Assembler		150	250	25	50	—	—	—
Debug		175	250	25	75	—	—	—
Extended Basic		—	—	25	100	20	50	200
Basic Precompiler		—	—	25	50	10	25	150
Sort/Merge		—	—	25	75	20	35	150
Utilities		—	Inc	25	75	10	25	150
Diagnostics		—	—	25	75	—	—	—
Text Processor		150	250	25	75	20	35	150
68000 X-ASMB on 6809		—	—	25	250	20	35	300
Pascal		—	—	50	200	25	50	300
Rel ASMB/Linking Loader		—	—	25	150	20	35	175
6800 X-ASMB on 6809		—	—	—	100	—	—	—
Cobol		—	—	—	—	30	75	750
Fortran 77		—	—	—	—	35	65	450

Software by Microware Systems Corp.	Run-Time Package	Update	Source	Man. Only	Object w/Man.
OS-9™ Level One Operating System		75.00	400.00	40.00	200.00
OS-9™ Level Two Operating System		75.00	N/A	40.00	500.00
BASIC09™	100.00	75.00	N/A	25.00	200.00
OS-9™ Micro Text Editor			300.00	15.00	125.00
OS-9™ Interactive Assembler			300.00	10.00	125.00
OS-9™ Interactive Debugger (Disk version)			100.00	10.00	50.00
CIS Cobol Compiler	400.00	50.00	N/A	80.00	900.00
Pascal Compiler	100.00	100.00	N/A	40.00	400.00
"C" Compiler		100.00	N/A	40.00	400.00

Microware yearly support service ($200.00 for OS-9 Level 2) 75.00

Special Software

2K 6809 MICROBUG	30.00	4K 6809 HUMBUG	75.00
2K 6800 HUMBUG	40.00	4K 6800 HUMBUG	65.00

Other HUMBUG versions including video versions are available.

Spell'n Fix by Peter Stark	178.58	Written Spell by Peter Stark	75.11
All-In-One, Spell'n Fix, and Written Spell package			250.00
Dynamite Disassembler			60.00
SUPER SLEUTH Disassembler System ($101.00 for DS-9 version)			99.00

SD/DD DISK DRIVES	1 head Tandon	2 heads Tandon	2 heads CDC	1 head MPI	2 heads MPI
30 day guarantee					
5-1/4", 40 tracks	225.00	300.00	300.00	250.00	325.00
5-1/4", 80 tracks	300.00	375.00	375.00	325.00	400.00

MPI or CDC Service Manual (Specify 40 or 80 track) 25.00
8", 77 tracks DS/DD Qume QT-8 $550.00 Remex (Special) 350.00

SPECIALS

* U.S. Robotics 300/1200 Baud auto dial/auto answer modem					499.00
* Same as above but without self test and diagnostics					399.00
* U.S. Robotics 1200 baud direct connect auto answer modem					349.00
* Hazeltine 1420 CRT terminal (new)					375.00
* SSB SFD Floppy Disk Controller (Version 3) Run FLEX or SSB DOS					100.00
* SWTPC 4K Memory $15.00	MP-B	$40.00	MP-A with MP-C		35.00
* High speed tape reader 50.00	AC-30	$40.00	MP-S		60.00
* 300 Baud acoustic modem 129.00					
* 10 ELEKTRA DD 8" Disks 38.00	10 5" DD Disks in hard box				25.00
* TI 810 Printer, w/lower case and full vertical forms control					1200.00
* Microtime II Calender and Clock Board (Assembled)					60.00

GIMIX CLEARANCE SALE	LIST	OUR PRICE		LIST	OUR PRICE
05 6809 Plus CPU Bd.	576.05	475.00	6800 CPU Board	224.03	100.00
Cable (Ser or Par I/O)	24.95	20.00	Motherboard		200.00
Double disk reg. card	68.22	50.00	8 Port Serial I/O Bd.	318.46	250.00
32K memory board		175.00	828 control w/GMX Flex	328.28	270.00
64K memory board	478.67	450.00	56K memory board		425.00
80 X 24 Video Boards	398.76	250.00	Single prt ser, 1 cable	113.36	90.00
64 x 16 Video Boards	198.71	100.00	Dual prt par, 2 cables	138.32	110.00
16K Mem Bds. w/cmn reg		145.00	4K PPD PROM Bd. and burner		100.00
93L422 RAM chips (2 needed for GIMIX DAT)				each	20.00

SWTPC

DC-4 5-1/4" Disk Controller	230.00	SWTPC 6809 FLEX™ Disk & Man	35.00
MP-S2 Dual Port Serial	120.00	MP-L2 Dual Port Parallel	120.00
MP-N Calculator Board (kit)	54.95	MP-N (assembled)	92.00
MP-R 2716 Eprom Programmer	114.50	MP-09 2MHz 6809 CPU Board	295.00

Smoke Signal Broadcasting

DCB-4A Double Density Controller Board for 5" and 8" with DOS	549.00
SSB DOS (Specify 6800 or 6809, BFD or DCB-4A, 5" or 8")	75.00
SE92/SA92-5 (6809 Edit/Assm for DOS)	69.95
SSB Monitor (Specify 6800/6809, 6800/6809)	75.00
SSB version of FLEX™ (without Editor and Assembler)	150.00
LMB-1A Motherboard	399.00
SCB-09 6809 CPU Board	399.00
PAR-1 Dual Port Parallel Board	89.00
SER-2 Dual Port Serial Board with 2 Cables	129.00
Chief 90 64K Computer System	2195.00

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits. AAA Chicago Computer Center does warranty and maintain service for our assembled boards. The customer should carefully take into consideration the small differential separating our kit and assembled prices when making his choice of purchase.

We have introduced our line of computer equipment with the purpose of offering the highest quality of components possible at affordable prices. These products are intended for OEM applications where it is the responsibility of the purchaser to integrate these components with suitable memory, disk controllers, drivers, and software along with I/O terminals to form working computer systems.

to make source programs cross-compatible. If they are completely successful, Step 4 would be unnecessary. You would, however, need the comparable compiler from the same vendor on both computers.

Good Luck!

Phil

"PK" PROGRAMABLE KEYBOARDS

BY

JOSEPH D CONDON

8101 ALPINE DRIVE
DES MOINES, IOWA 50322
TELEPHONE 515-278-4581

HAVE YOU EVER NOTICED WHILE OPERATING YOUR COMPUTER, YOU FIND YOURSELF ENTERING THE SAME COMMANDS, PHRASES, OR SEQUENCE OF KEY STROKES OVER AND OVER AGAIN? AFTER ALL, COMPUTERS ARE INTENDED TO PERFORM REPETITIVE TASKS QUICKLY AND EFFICIENTLY FOR THEIR USERS, SO WHY NOT TAKE ADVANTAGE OF THAT FACT? THE FOLLOWING PROGRAM (WRITTEN IN FLEX09 ASSEMBLER) DOES JUST EXACTLY THAT.

THE PROGRAM "PK" WILL ALLOW THE USER TO PROGRAM ANY TERMINAL KEY (HEX 40-5F) TO REPRESENT ANY TEXT STRING (UP TO 255 CHARACTERS, WITH EMBEDED CARRIAGE RETURNS REPRESENTED BY THE "\" CHARACTER). ONCE A KEY IS PROGRAMED, ITS REPRESENTED TEXT STRING WILL BE RETURNED AND DISPLAYED WHEN EVER THAT KEY IS ENTERED AS A CONTROL CHARACTER. THE USER MAY PROGRAM ANY NUMBER OF KEYS WITHIN THE SPCIFIED RANGE. ANY PROGRAMED KEY MAY BE RE-PROGRAMED AT ANY TIME TO REPRESENT A DIFFERENT TEXT STRING IF DESIRED. ALSO ANY PROGRAMED KEY MAY BE DE-PROGRAMED AT ANY TIME. IF THE USER WISHES, ALL PROGRAMED KEY MAY BE DEPROGRAMED WITH ONE COMMAND. THE USER MAY ALSO INQUIRE AS TO WHICH KEYS ARE CURRENTLY PROGRAMED WITH WHAT TEXT, AND THE MOUNT OF MEMORY BEING USED BY THE RESIDENT "PK" ROUTINE AND KEY TABLE ENTRYS. THE "PK" UTILITY MODIFIES THE "FLEX09" INCH2 VECTOR TO POINT TO THE RESIDENT "PK" ROUTINE WHICH IN TURN USES THE ORIGINAL INCH2 VECTOR FOR INPUT. THIS PROVIDES COMPATIBILITY WITH ALL I-O ROUTINES, CUSTOM OR STANDARD. "PK" WILL WORK PROPERLY WITH ALL PROGRAMS OR ROUTINES WHICH EVENTUALLY USE THE INCH2 ROUTINE FOR INPUT. "PK" IS INTENDED TO BE COMPATIBLE WITH ALL "FLEX09" SYSTEMS USING A TERMINAL KEYBOARD THAT IS CAPABLE OF GENERATING CONTROL CHARACTERS FOR THE KEYS HEX 40-5F, SUCH AS THE "ADM3" TERMINAL.

THE "PK" PROGRAM CONSISTS OF TWO PARTS. THE FIRST PART IS THE ACTUAL "PK" PROGRAM WHICH IS EXECUTED EACH TIME YOU ENTER THE "PK" COMMAND. THE SECOND PART IS A ROUTINE THAT GETS PERMANENTLY INSTALLED AT THE TOP OF USER MEMORY. WHENEVER THE "PK" PROGRAM IS EXECUTED, IT FIRST CHECKS TO SEE IF THE ROUTINE POINTED TO BY THE SYSTEMS INCH2 VECTOR (LOCATED AT $CD0C+1) IS THE SAME AS THE PKINCH ROUTINE. IF IT IS, THE PROGRAM THEN PERFORMS THE FUNCTION SPECIFIED BY THE INITIATING MESSAGE. IF THE ROUTINE POINTED TO BY THE SYSTEMS INCH2 VECTOR IS NOT THE SAME AS THE PKINCH ROUTINE, THE PROGRAM THEN MOVES THE PKINCH ROUTINE TO ITS RESIDENT LOCATION AT THE TOP OF USER MEMORY. THE PROGRAM THEN ADJUSTS THE MEMEND VALUE (LOCATION $CC2B) TO POINT TO THE LOCATION JUST BELOW THE RESIDENT PKINCH ROUTINE. THE SYSTEMS INCH2 VECTOR VALUE IS THEN MOVED TO A LOCATION WITHIN THE PKINCH ROUTINE, AND THE SYSTEMS INCH2 VECTOR LOCATION IS CHANGED TO POINT TO THE RESIDENT PKINCH ROUTINE.

AFTER INSURING THAT THE PKINCH ROUTINE IS RESIDENT IN MEMORY, THE PROGRAM THEN PERFORMS THE FUNCTION SPECIFIED BY THE INITIATING MESSAGE. THERE ARE THREE TYPES OF FUNCTIONS.THE FIRST, INITIATED BY THE SYNTAX "PK", WILL LIST ON THE OUTPUT DEVICE ALL THE KEYS THAT ARE CURRENTLY PROGRAMED ALONG WITH THEIR RESPECTIVE TEXT. IT WILL ALSO LIST THE TOTAL AMOUNT OF MEMORY BEING USED BY ALL THE KEY PROGRAMS AND THE RESIDENT PKINCH ROUTINE.

THE SECOND FUNCTION INITIATED BY THE SYNTAX "PK" <KEY>"="<TEXT>, WILL CREATE A KEY PROGRAM FOR THE KEY SPECIFIED BY THE <KEY> VALUE CONSISTING OF THE TEXT SPECIFIED BY THE <TEXT> STRING. THE "=" CHARACTER IS A REQUIRED PART OF THE SYNTAX. WHEN CREATING A KEY PROGRAM, THE "PK" PROGRAM FIRST CALCULATES THE LENGTH OF THE TEXT STRING. IT THEN ADDS 4 TO THIS VALUE, 2 BYTES FOR THE NEXT KEY PROGRAM POINTER 1 BYTE FOR THE KEY CHARACTER, AND ONE BYTE FOR THE TEXT LENGTH. AFTER THE TOTAL AMOUNT OF MEMORY REQUIRED FOR THE KEY PROGRAM IS DETERMINED, THE KEY PROGRAM IS THEN INSERTED AT THE TOP OF USER MEMORY. THE SYSTEMS MEMEND VALUE IS THEN ADJUSTED TO POINT JUST BELOW THE KEY PROGRAM. THE PKINCH ROUTINES KEY TABLE POINTER IS THEN SET TO POINT TO THE KEY PROGRAMS LOCATION, AND THE KEY PROGRAMS POINTER IS SET TO POINT TO THE LOCATION THAT THE PKINCH ROUTINES POINTER ORIGINALLY POINTED TO. THIS PROCEDURE MAINTAINS A LINKED LIST IN MEMORY JUST ABOVE THE SYSTEMS USER MEMORY. THE LAST ENTRY IN THE LINKED LIST WILL ALWAYS CONTAIN A POINTER VALUE OF ZERO. AS ADDITIONAL KEYS ARE PROGRAMED, THEIR PROGRAMS ARE ALWAYS INSERTED AT THE TOP OF THE LIST.

THE LAST FUNCTION OF THE "PK" PROGRAM INITIATED BY THE SYNTAX "PK -" WILL REMOVE ALL EXISTING KEY PROGRAMS BY SETING THE SYSTEMS MEMEND VALUE TO A LOCATION JUST BELOW THE PKINCH ROUTINE AND CLEARING THE PKINCH ROUTINES KEY TABLE POINTER. ONE FACT TO BE AWARE OF IS THAT IF AFTER FIRST EXECUTING THE "PK" PROGRAM YOU INSTALL SOME OTHER ROUTINE AT THE TOP OF USER MEMORY AND THEN ADJUST THE MEMEND VALUE, THAT ROUTINE WILL BECOME PART OF USER MEMORY AND WILL PROBABLY GET OVERWRITTEN AFTER USING THE "PK -" FUNCTION.

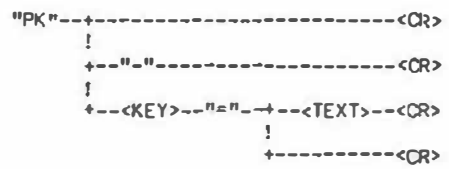
THE SECOND PART OF THE PROGRAM IS THE ACTUAL PKINCH ROUTINE. THIS ROUTINE IS INSTALLED AT THE TOP OF USER MEMORY BY THE "PK" PROGRAM AS DESCRIBED EARLIER. EACH TIME THE SYSTEM USES ITS INCH2 ROUTINE IT ACTUALLY USES THE PKINCH ROUTINE FIRST.THE PKINCH ROUTINE FIRST CHECKS TO SEE IF A KEY PROGRAM IS IN PROGRESS,THAT IS, IF A SERIES OF CHARACTERS ARE CURRENTLY BEING PROVIDED TO THE SYSTEM BY THE PKINCH ROUTINE. IF A KEY PROGRAM IS ACTIVE, THE ROUTINE THEN PROVIDES THE SYSTEM WITH THE NEXT KEY PROGRAM CHARACTER AFTER CALLING THE SYSTEMS OUTCH ROUTINE. IF NO KEY PROGRAM IS ACTIVE, THE PKINCH ROUTINE CALLS THE SYSTEMS ORIGINAL INCH2 ROUTINE. AFTER RECEIVING A CHARACTER FROM THE INCH2 ROUTINE, PKINCH THEN SCANS THROUGH ITS KEY PROGRAM LIST TO SEE IF THE CHARACTER ENTERED HAS BEEN PREVIOUSLY PROGRAMED. IF NOT, THE CHARACTER IS RETURNED TO THE SYSTEM. IF THE CHARACTER HAS BEEN PROGRAMED WITH A NON ZERO TEXT LENGTH, THE PKINCH ROUTINE MARKS THE KEY PROGRAM AS ACTIVE AND RETURNS THE FIRST CHARACTER OF THAT KEY PROGRAMS TEXT TO THE SYSTEM. IF THE CHARACTER HAS BEEN PROGRAMED WITH A ZERO TEXT LENGTH ("PK <KEY>="), THE PKINCH ROUTINE SIMPLY RETURNS THE CHARACTER TO THE SYSTEM. PROGRAMING A KEY WITH A ZERO TEXT LENGTH WILL EFFECTIVELY DE-PROGRAM THAT KEY. SINCE ALL KEY PROGRAMS ARE ENTERED AT THE TOP OF THE KEY PROGRAM LIST, ONLY THE MOST CURRENT PRO-

GRAM FOR ANY GIVEN KEY WILL BE USED BY THE PKINCH ROUTINE.

ONE LAST COMMENT; "FLEX09" MAKES USE OF SOME PREDEFINED CONTROL CHARACTERS FOR SPECIAL FUNCTIONS. THE CONTROL "X" CHARACTER FOR EXAMPLE IS USED TO CANCEL INPUT OF A COMMAND LINE, THE CONTROL "H" CHARACTER WILL GENERATE A BACK SPACE, AND SO ON. IF YOU SHOULD PROGRAM ONE OF THESE SPECIAL FUNCTION KEYS, THE SYSTEM WILL NO LONGER RESPOND TO THAT KEY AS INTENDED. IF YOU RUN INTO THIS CONDITION, SIMPLY DE-PROGRAM THAT KEY WITH THE COMMAND "PK <KEY>=". ALTHOUGH THE CONTROL "M" KEY FALLS WITHIN THE RANGE OF PROGRAMABLE CHARACTERS,"PK" WILL NOT ALLOW YOU TO PROGRAM IT. THE CONTROL "M" CHARACTER IS THE SAME AS A CARRIAGE RETURN AND CHANGING ITS DEFINITION WOULD RENDER YOUR SYSTEM USELESS UNTILL YOU RE-BOOT.

THE FOLLOWING SYNTAX DIAGRAM AND EXAMPLES WILL DEMONSTRATE THE USE OF THE "PK" PROGRAM.

SYNTAX DIAGRAM



<KEY> = ANY KEY FROM HEX 40 TO HEX 5F.
"M" KEY NOT INCLUDED.

<TEXT> = UP TO 255 ASCII CHARACTERS.
"\" REPRESENTS A <CR>

EXAMPLES

```
*PK"
*PK -"
*PK C=CAT\"
*PK C=CAT 1\"
*PK A=ASMB 1.TEST,1.TEST.CMD +SYN1\
*PK E=EDIT 1.TEST\Y"
    (NOTE THAT THE "Y" WILL BE THE RESPONSE
    TO THE )
    (EDITORS PROMPT, "DELETE BACKUP FILE
    Y/N")
```

```
OPT     PAG
TTL     PK
STTL    PROGRAMABLE KEYBOARD
```

```
*****************************************
*                                       *
*                                       *
*                "PK"                   *
*                                       *
*                                       *
*        PROGRAMABLE KEYBOARD           *
*                                       *
*                                       *
*           JOE CONDON                  *
*           12/23/83                    *
*                                       *
*                                       *
*****************************************
```

```
*****************************************
*                                       *
*            FLEX EQUATES               *
*                                       *
*****************************************

VERNO   EQU     1           VERSION NUMBER

EOT     EQU     $04         ASCII EOT CHARACTER
CR      EQU     $0D         ASCII CR CHARACTER

UTIL    EQU     $C100       UTILITY AREA
LBPNT   EQU     $CC14       LINE BUFFER POINTER
MEMEND  EQU     $CC28       MEMORY END
WARMS   EQU     $CD03       FLEX RE-ENTRY POINT
INCH2   EQU     $CD0C       SYSTEM INCH2 ROUTINE
OUTCH   EQU     $CD0F       SYSTEM OUTCH ROUTINE
PUTCHR  EQU     $CD18       PRINT ASCII CHARACTER
PSTRNG  EQU     $CD1E       PRINT ASCII CHAR STRING
PCRLF   EQU     $CD24       PRINT CARR RETURN LINE FEED
NXTCH   EQU     $CD27       GET NEXT BUFFER CHARACTER
OUTDEC  EQU     $CD39       PRINT DECIMAL VALUE

INCH2V  EQU     INCH2+1     SYSTEM INCH2 VECTOR
PKISIZ  EQU     PKIVAR-PKINCH  PKINCH ROUTINE SIZE

        PAG
*****************************************
*                                       *
*          START OF PROGRAM             *
*                                       *
*****************************************

        ORG     UTIL        PROGRAM ORIGIN

START   BRA     PK          BRANCH AROUND VERSION NO
        FCB     VERNO       VERSION NUMBER

PK      LDX     INCH2V      GET SYSTEM INCH2 VECTOR
        LDY     #PKINCH     POINT TO PKINCH ROUTINE

PK1     LDA     0,X+        GET RESIDENT INCH BYTE
        CMPA    0,Y+        COMPARE TO PKINCH BYTE
        BNE     PK2         INSTALL PKINCH ROUTINE
        CMPY    #PKIVAR     TEST FOR PKINCH END
        BNE     PK1         COMPARE NEXT BYTE
        BRA     PK4         PKINCH ROUTINE RESIDENT

PK2     LDD     MEMEND      GET MEMORY END
        SUBD    #PKISIZ+7   CALCULATE PKINCH LENGTH
        STD     MEMEND      SET NEW MEMORY END
        ADDD    #1          POINT TO PKINCH ROUTINE
        TFR     D,X         TRANSFER D TO X REG
        LDY     #PKINCH     POINT TO PKINCH ROUTINE

PK3     LDA     0,Y+        GET PKINCH ROUTINE BYTE
        STA     0,X+        STORE PKINCH ROUTINE BYTE
        CMPY    #PKIVAR     TEST FOR END OF ROUTINE
        BNE     PK3         MOVE NEXT BYTE
        LDD     INCH2V      GET SYSTEM INCH2 VECTOR
        STD     0,X++       SET ORIGINAL INCH2 VECTOR
        CLR     0,X+        CLEAR KTPNT UPPER
        CLR     0,X+        CLEAR KTPNT LOWER
        CLR     0,X+        CLEAR PKCPNT UPPER
        CLR     0,X+        CLEAR PKCPNT LOWER
        CLR     0,X         CLEAR PKCCNT
        LDD     MEMEND      GET MEMEND POINTER
        ADDD    #1          POINT TO PKINCH ROUTINE
        STD     INCH2V      SET NEW SYSTEM INCH2 VECTOR

PK4     JSR     NXTCH       GET NEXT LINE BUFFER CHAR
        CMPA    #CR         TEST FOR END OF LINE
        BNE     PK9         CHECK NEXT CHARACTER
        JSR     PCRLF       PRINT CR & LF
        CLR     PKCHAR      SET CHAR TO FIRST CTRL CHAR

PK5     LDD     INCH2V      GET SYSTEM INCH2 VECTOR
        ADDD    #PKISIZ+2   CALC POINTER LOC
```

```
        TFR     D,X         MOVE D REG TO X REG

PK6     LDX     0,X         POINT TO TABLE ENTRY
        BEQ     PK8         GET NEXT KEY CHARACTER
        LDA     PKCHAR      GET KEY CHARACTER
        CMPA    2,X         COMPARE TO TABLE CHAR
        BNE     PK6         GET NEXT TABLE ENTRY
        TST     3,X         CHECK FOR ZERO LENGTH
        BEQ     PK8         GET NEXT KEY CHARACTER
        JSR     PCRLF       PRINT CR & LF
        LDA     PKCHAR      GET KEY CHARACTER
        ADDA    #$40        CONVERT CHAR TO ASCII
        JSR     PUTCHR      PRINT ASCII CHAR
        LDA     #'=         GET EQUAL CHAR
        JSR     PUTCHR      PRINT ASCII CHAR
        LDB     3,X         GET TEXT LENGTH
        BEQ     PK8         GET NEXT KEY CHARACTER
        LEAX    4,X         POINT TO TEXT

PK7     LDA     0,X+        GET TEXT CHAR
        JSR     PUTCHR      PRINT ASCII CHAR
        DECB                DECREMENT TEXT LENGTH
        BNE     PK7         PRINT NEXT TEXT CHAR

PK8     INC     PKCHAR      NEXT CTRL KEY
        LDA     PKCHAR      GET CTRL CHARACTER
        CMPA    #$1F        TEST FOR LAST CTRL CHAR
        BLE     PK5         LIST NEXT KEY PROGRAM
        JSR     PCRLF       PRINT CR & LF
        LDX     #MESS04     POINT TO MESSAGE
        JSR     PSTRNG      PRINT STRING
        LDD     INCH2V      GET PKINCH VECTOR
        ADDD    #PKISIZ+7   ADD PKINCH LENGTH
        SUBD    MEMEND      CALCULATE MEMORY USED
        SUBD    #1          ADJUST FOR ABSOLUTE SIZE
        STD     TXTLEN      STORE VALUE FOR PRINTING
        LDX     #TXTLEN     POINT TO VALUE
        CLRB                SET SUPRESSION FLAG
        JSR     OUTDEC      PRINT DECIMAL VALUE
        LBRA    PK19        PRINT CRLF THEN END

PK9     CMPA    #'-         TEST FOR REMOVE CHAR
        BNE     PK10        PROGRAM KEY
        LDD     INCH2V      GET SYSTEM INCH2 VECTOR
        ADDD    #PKISIZ+2   CALC POINTER LOC
        TFR     D,X         MOVE D REG TO X REG
        CLR     0,X+        CLEAR KTPNT UPPER
        CLR     0,X         CLEAR KTPNT LOWER
        LDD     INCH2V      GET SYSTEM INCH2 VECTOR
        SUBD    #1          CALCULATE MEMORY END
        STD     MEMEND      SET NEW MEMEND
        LDX     #MESS03     POINT TO MESSAGE
        LBRA    PK18        PRINT MESSAGE THEN END

PK10    CMPA    #$40        TEST FOR KEY TO LOW
        BGE     PK11        TEST FOR KEY TO HIGH
        LDX     #MESS00     POINT TO MESSAGE
        BRA     PK18        REPORT ERROR THEN END

PK11    CMPA    #$5F        TEST FOR KEY TO HIGH
        BLE     PK12        TEST FOR M KEY
        LDX     #MESS00     POINT TO MESSAGE
        BRA     PK18        REPORT ERROR THEN END

PK12    CMPA    #'M         TEST FOR M KEY
        BNE     PK13        PROGRAM KEY
        LDX     #MESS00     POINT TO MESSAGE
        BRA     PK18        REPORT ERROR THEN END

PK13    SUBA    #$40        CONVERT KEY TO CTRL CHAR
        STA     PKCHAR      STORE CHARACTER
        JSR     NXTCH       GET NEXT LINE BUFFER CHAR
        CMPA    #'=         TEST FOR EQUAL CHARACTER
        BEQ     PK14        FORMAT OK
        LDX     #MESS01     POINT TO MESSAGE
        BRA     PK18        REPORT ERROR THEN END

PK14    LDD     LBPNT       GET LINE BUFFER POINTER
        STD     TXTLEN      SAVE LINE BUFFER POINTER

PK15    JSR     NXTCH       GET NEXT LINE BUFFER CHAR
        CMPA    #CR         TEST FOR END OF LINE
        BNE     PK15        GET NEXT CHARACTER
        LDD     LBPNT       GET LINE BUFFER POINTER
        SUBD    TXTLEN      CALCULATE TEXT LENGTH
        STD     TXTLEN      SAVE TEXT LENGTH
        LDD     LBPNT       GET LINE BUFFER POINTER
        SUBD    TXTLEN      CALCULATE ORIG POINTER
        STD     LBPNT       RESTORE LINE BUFFER POINTER
        LDD     MEMEND      GET MEMORY END
        SUBD    #4          SUBTRACT TABLE HEADER LEN
        SUBD    TXTLEN      SUBTRACT TEXT LENGTH
        STD     MEMEND      SET NEW MEMORY END
        ADDD    #1          POINT TO START OF HEADER
        TFR     D,X         MOVE D REG TO X REG
        LDD     INCH2V      GET SYSTEM INCH2 VECTOR

        ADDD    #PKISIZ+2   CALC TABLE POINTER LOC
        TFR     D,Y         MOVE D REG TO Y REG
        LDD     0,Y         GET FIRST TABLE ENTRY LOC
        STD     0,X         SET NEXT ENTRY POINTER
        STX     0,Y         SET NEW FIRST ENTRY LOC
        LEAX    2,X         POINT TO HEADER CHARACTER
        LDA     PKCHAR      GET KEY CHARACTER
        STA     0,X+        SET HEADER CHARACTER
        LDD     TXTLEN      GET TEXT LENGTH
        STB     0,X+        SET HEADER TEXT LENGTH

PK16    BEQ     PK17        NO MORE CHARACTERS
        JSR     NXTCH       GET NEXT LINE BUFFER CHAR
        STA     0,X+        STORE CHAR IN RECORD TEXT
        DECB                DECREMENT CHARACTER COUNT
        BRA     PK16        MOVE NEXT CHARACTER

PK17    LDX     #MESS02     POINT TO MESSAGE

PK18    JSR     PCRLF       PRINT CR & LF
        JSR     PSTRNG      PRINT STRING

PK19    JSR     PCRLF       PRINT CR & LF
        JMP     WARMS       RETURN TO FLEX

        PAG
*****************************************************
*                                                   *
*                   DATA STORAGE                    *
*                                                   *
*****************************************************

MESS00  FCC     'ILLEGAL KEY SELECTION',EOT
MESS01  FCC     'MISSING = CHARACTER',EOT
MESS02  FCC     'KEY PROGRAMED',EOT
MESS03  FCC     'KEY PROGRAMS REMOVED',EOT
MESS04  FCC     'MEMORY USED = ',EOT

*****************************************************
*                                                   *
*                VARIABLE STORAGE                   *
*                                                   *
*****************************************************

PKCHAR  RMB     1           CHAR TO BE PROGRAMED
TXTLEN  RMB     2           KEY TEXT LENGTH

        PAG
*****************************************************
*                                                   *
*                                                   *
*    PROGRAMABLE KEYBOARD INPUT CHAR ROUTINE        *
*                                                   *
*                                                   *
*           KEY TABLE RECORD FORMAT                 *
*           -----------------------                 *
*           NEXT ENTRY    2 BYTES                   *
*           CHARACTER     1 BYTE                    *
*           TEXT LENGTH   1 BYTE                    *
*           TEXT          0-255 BYTES               *
```

```
*                                                     *
*                                                     *
*******************************************************

PKINCH  PSH     X,U         SAVE X & U REGISTERS
        LEAU    PKIVAR,PCR  POINT TO PKINCH VARIABLES
        LDX     2,U         GET KEY TABLE POINTER
        BNE     PKI1        TABLE ENTRYS PRESENT
        JSR     [0,U]       USE SYSTEM INCH2 ROUTINE
        BRA     PKI7        RESTORE REGISTERS & RETURN

PKI1    TST     6,U         TEST FOR SEQUENCE ACTIVE
        BEQ     PKI3        SEQUENCE NOT ACTIVE
        LDX     4,U         GET CHARACTER POINTER
        LDA     0,X+        GET NEXT CHARACTER
        STX     4,U         STORE CHARACTER POINTER
        DEC     6,U         DECREMENT CHARACTER COUNTER
        CMPA    #'\         TEST FOR DEFINED CR KEY
        BNE     PKI2        OUTPUT CHARACTER
        LDA     #CR         SUBSTITUTE CR CHARACTER

PKI2    JSR     OUTCH       OUTPUT CHARACTER
        BRA     PKI7        RESTORE REGISTERS & RETURN

PKI3    JSR     [0,U]       USE SYSTEM INCH2 ROUTINE

PKI4    CMPA    2,X         TEST FOR KEY MATCH
        BEQ     PKI5        MATCH FOUND
        LDX     0,X         GET NEXT TABLE ENTRY
        BNE     PKI4        TEST NEXT TABLE ENTRY
        BRA     PKI7        RESTORE REGISTERS & RETURN

PKI5    LEAX    3,X         POINT TO TABLE TEXT LENGTH
        TST     0,X         TEST TABLE TEXT LENGTH
        BEQ     PKI7        NO CHARACTERS
        LDA     0,X+        GET TABLE TEXT LENGTH
        DECA                DECREMENT TEXT LENGTH
        STA     6,U         SET CHARACTER COUNTER
        LDA     0,X+        GET TABLE CHARACTER
        STX     4,U         STORE CHARACTER POINTER
        CMPA    #'\         TEST FOR DEFINED CR KEY
        BNE     PKI6        OUTPUT CHARACTER
        LDA     #CR         SUBSTITUTE CR CHARACTER

PKI6    JSR     OUTCH       OUTPUT CHARACTER

PKI7    PUL     X,U         RESTORE X & U REGISTERS
        RTS                 RETURN FROM SUBROUTINE

PKIVAR  RMB     2           ORIGINAL SYSTEM INCH2 VECTOR
        RMB     2           KEY TABLE POINTER
        RMB     2           CHARACTER POINTER
        RMB     1           CHARACTER COUNTER

*******************************************************

        END     START
```

EPSON MX-80 PARALLEL PRINTER DRIVER (PC.CMD)

Tom J. Harmon
15418 Diana Lane
Houston, TX 77062
Phone 713 480-6075

This article describes a parallel print command (PC.CMD) written for the EPSON MX-80 printer with Graftrax. The routine is a modified version of SWTPC's P.CMD routine and contains some of the coding presented by Kenneth Drexler in his FLEXIBLE P SYS Article published in the March 1983 issue of 68 Micro Journal. The print command parameters are exactly the same as those used by the SWPTC P.CMD routine. For example, to output to port 7A, use the command PC#7-A. The default port assignment has been changed to port 7B for compatibility with OS-9 software. You may reset the default side to the A port by changing the SIDE parameter to zero.

The PRINT.SYS routine is no longer being used used by SWPTC. The P.CMD supplied with the SWPTC version of Flex includes a relocatable printer driver. I placed additional logic in the relocatable driver which begins at $C300. This routine now tests for the presence of a two character code of the form up arrow D where D can have one of the numeric values 0 thru 9 or an alphabetic value A thru J. These printer codes are imbedded in the text file being printed and tell the Epson printer driver to send a one or two byte sequence to the Epson printer to perform the appropriate action such as form feed, switching to compressed print, etc.

The printer codes are listed below:

NOTE: 'UP' indicates 'up arrow':

'UP'0 backspace
'UP'1 start underline - not implemented
'UP'2 stop underline - not implemented
'UP'3 start emphasized
'UP'4 end emphasized
'UP'5 start double width
'UP'6 end double width
'UP'7 BELL
'UP'8 not used
'UP'9 not used
'UP'A line feed
'UP'B home head
'UP'C form feed
'UP'D carriage return
'UP'E start italics
'UP'F end italics
'UP'G start double strike
'UP'H end double strike
'UP'I start compressed print
'UP'J end compressed print

Please remember to use relocatable assembly code if you plan to make any additional modifications to the program I have supplied. This program doesn't require the "hidden memory" that is mentioned in Kenneth Drexler's article. The left/right margin control was also omitted since this feature is available from the TSC Text Processor. The default page size is defined to be 60 lines per page. The actual page size expected by the Epson printer is 66 lines so this allows room for three lines at the top and bottom of the page. Set the top of margin three lines down from the top of page for best results.

This driver assumes the Epson printer has been placed at the top of form each time it is called since the line counter (LINCT) is initially set to zero. The initialization coding to issue a form feed wasn't included since I didn't want to waste paper for short printouts like directory listings where several print commands might all fit on a single page. You can force the driver to issue a form feed by including an up arrow C command on the first line of the file being printed.

This routine is being used with the SWPTC MP-L2 dual-port parallel interface card. It should work equally well with the MP-L card and should support both the 69/A series system and the S09 systems. Please let me know if you make additional modifications to this driver.

- - -

```
                        OPT   PAG
                        TTL   EPSON MX-80 PRINT COMMAND PC.CMD
                * PC.CMD
                *
                * This routine is a modified version
                * of SWTPC's Flex utility P.CMD.
                *
                * MODIFIED BY Tom J. Hanson
                *          15418 Diana Lane
                *          Houston, TX 77062
                *          Aug 2, 1983
                *          Phone 713 480-6075
                *
CCD9    TTYPS   EQU   $CCD9
CC11    LSTTRM  EQU   $CC11
CC28    CMDFLG  EQU   $CC28
CC2B    MEMEND  EQU   $CC2B
CC33    CPUTYP  EQU   $CC33
CC35    PRTADR  EQU   $CC35
CC37    PRTLNG  EQU   $CC37
CC39    PRTDVC  EQU   $CC39
CCD0    PTERM   EQU   $CCD0
CCDB    PCMX    EQU   $CCDB
CCE4    POUT    EQU   $CCE4     PRINTER CHARACTER OUTPUT ROUTINE
CCFC    PWCFLG  EQU   $CCFC     ACTIVE SPOOLING PROCESS FLAG
CD03    WARMS   EQU   $CD03
CD0F    OUTCH   EQU   $CD0F
CD1E    PSTRNG  EQU   $CD1E
CD27    MITCH   EQU   $CD27
CD48    INDEC   EQU   $CD48
CD48    OOCMND  EQU   $CD48
                *
                * ASCII CODE EQUATES
                *
0004    EOT     EQU   $04
                *
                *    EXTERNAL EQUATES
                *
D3E0    DUMMY   EQU   $D3E0     DUMMY RTS INSTR USED BY RM
D3ED    T_OFF   EQU   $D3ED     TIMER OFF ROUTINE
D3EF    T_ON    EQU   $D3EF     TIMER ON ROUTINE
D3F1    T_INIT  EQU   $D3F1     TIMER INITIALIZE ROUTINE
                *
C100            ORG   $C100
C100 20 05      PCMD  BRA   ENTRY     P.CMD ENTRY POINT
C102 83 2E 81         FCB   $83,$2E,$81 VERSION NUMBER
C105 3A 86            FCB   $3A,$86
C107 B6 CC09    ENTRY LDA   TTYPS     SAVE FLEX CONSTANTS
C10A F6 CC28          LDB   CMDFLG
C10D ED 8D 01E6       STD   ATEMP,PCR
C111 FC CC2B          LDD   MEMEND    SAVE END-OF-MEMORY LOC.
C114 E0 8D 01E1       STD   FLIEND,PCR
C118 FC CC43          LDD   $CC43
C11B ED 8D 010C       STD   BTEMP,PCR
C11F 7D CCFC          TST   $CCFC     TEST FOR PRINTER OVERLAY
C122 1026 012C        LBNE  WMS61
C126 CC D3E0          LDD   #DUMMY
C129 FD D3F1          STD   T_INIT
C12C FD D3EF          STD   T_ON
C12F FD D3ED          STD   T_OFF
C132 8E C300          LDX   #$C300    ADDRESS OF RELOCATABLE DRIVER
C135 EC 81            LDD   0,X++
C137 1027 011C        LBEQ  WMS62     LENGTH IS ZERO
C13B 10BE CC35        LDY   PRTADR
C13F 27 06            BEQ   LC147
C141 1083 CC37        CMPD  PRTLNG
C145 23 07            BLS   LC14E
C147 17 00C0    LC147 LBSR  LC20A     RELOCATE DRIVER
C14A 1023 0114        LBLS  WMS64
C14E 17 00CB    LC14E LBSR  LC21C
C151 17 00D7          LBSR  LC228
C154 B6 CC11          LDA   LSTTRM
C157 81 23            CMPA  #'#       POUND SIGN
C159 27 05            BEQ   LC160     YES
C15B 17 0087          LBSR  LC1E5
C15E 20 10            BRA   LC170
C160 BD CD48    LC160 JSR   INDEC     CONVERT INTO DECIMAL
C163 1F 10            TFR   X,D
C165 1083 0000        CMPD  #$0000
C169 1024 00F0        LBCC  WMS63     INVALID PORT NO.
C16D 17 00BB          LBSR  LC1F8
C170 B6 CC11    LC170 LDA   LSTTRM
C173 81 2D            CMPA  #$2D
C175 27 04            BEQ   LC17B
C177 81 2F            CMPA  #'/
C179 26 1B            BNE   LC196
C17B BD CD27    LC17B JSR   MITCH     GET PORT ADDRESS
C17E 84 3F            ANDA  #$3F
C180 80 41            SUBA  #'A       EXPECT A THRU F
C182 1025 00D7        LBCS  WMS63     INVALID PORT NO.
C186 81 06            CMPA  #6
C188 1024 00D1        LBCC  WMS63     INVALID PORT NO.
C18C 8A 80            ORA   #$80
C18E A7 22            STA   2,Y
C190 BD CD27          JSR   MITCH
C193 B7 CC11          STA   LSTTRM
C196 CC CCE4    LC196 LDD   #POUT     ADDRESS OF OUTCH VECTOR
C199 FD CD10          STD   OUTCH+1
C19C 86 7E            LDA   #$7E      JMP INSTRUCTION
C19E B7 CD0F          STA   OUTCH
C1A1 CC 0045          LDD   #$0045    LENGTH OF ROUTINE
C1A4 30 8C 0110       LEAX  LC2BB,PCR
C1A8 BD 60            BSR   LC20A     ALLOCATE MEMORY
C1AA 1023 00B4        LBLS  WMS64
C1AE 8D 6C            BSR   LC21C     MOVE RELOCATABLE CODE
C1B0 6E A4            JMP   0,Y
C1B2 34 36      LC1B2 PSHS  A,B,I,Y
C1B4 7D CCFC          TST   $CCFC     TEST FOR PRINTER OVERLAY
C1B7 26 03            BNE   LC1CC
C1B9 8E C300          LDX   #$C300    ADDRESS OF PRINTER DRIVER
C1BC EC 81            LDD   0,X++     LENGTH OF DRIVER
C1BE 27 0C            BEQ   LC1CC     ZERO
C1C0 10BE CC35        LDY   PRTADR    PRINTER PORT ADDRESS
C1C4 27 06            BEQ   LC1CC
C1C6 1083 CC37        CMPD  PRTLNG
C1CA 23 07            BLS   LC1D3
C1CC B6 39      LC1CC LDA   #$39      RETURN INSTRUCTION
C1CE B7 CCC0          STA   PINIT
C1D1 35 B6            PULS  A,B,I,Y,PC
C1D3 BD 47      LC1D3 BSR   LC21C
C1D5 8D 54            BSR   LC228
C1D7 FC CC39          LDD   PRTDVC
C1DA 26 02            BNE   LC1DE
C1DC BD 07            BSR   LC1E5
C1DE ED 84      LC1DE STD   0,Y
C1E0 BD CCC0          JSR   PINIT
C1E3 35 B6            PULS  A,B,I,Y,PC
C1E5 EC A4      LC1E5 LDD   0,Y
C1E7 2A 0F            BPL   LC1F8
C1E9 1083 E0B2        CMPD  #$E0B2
C1ED 26 13            BNE   LC202
C1EF B6 CC33          LDA   CPUTYP    CPU TYPE
C1F2 85 02            BITA  #2
C1F4 26 0C            BNE   LC202
C1F6 C6 07            LDB   #7        PARALLEL INTERFACE SLOT
C1F8 B6 CC33    LC1F8 LDA   CPUTYP
C1FB 84 04            ANDA  #4
C1FD 26 02            BNE   LC201
C1FF 86 10            LDA   #$10
C201 3D         LC201 MUL             BUILD PRINTER
C202 B6 E0      LC202 LDA   #$E0       PORT ADDRESS IN A
C204 EB A4            STB   0,Y
C206 F8 CC39          STB   PRTDVC    SAVE PRINTER ADDRESS
C209 39               RTS
C20A 34 06      LC20A PSHS  A,B       NUMBER OF BYTES TO ALLOCATE
C20C FC CC2B          LDD   MEMEND    PREVIOUS MEMORY END
C20F A3 E4            SUBD  0,S       SUBTRACT VALUE IN D REG
C211 34 01            PSHS  CC        SAVE CC REGISTER
C213 FD CC2B          STD   MEMEND    UPDATE MEMEND
C216 1F 02            TFR   D,Y       STARTING ADDRESS
C218 31 21            LEAY  1,Y       JMP ADDRESS
C21A 35 87            PULS  CC,A,B,PC RESTORE REGISTERS
C21C 34 36      LC21C PSHS  A,B,I,Y
C21E A6 80      LC21E LDA   0,I+      MOVE ARRAY FROM X TO Y
C220 A7 A0            STA   0,Y+      LENGTH IS IN D REGISTER
C222 5A               DECB
C223 26 F9            BNE   LC21E
C225 6A E4            DEC   0,S       DECREMENT LENGTH
C227 2A F5            BPL   LC21E
C229 35 B6            PULS  A,B,I,Y,PC
```

```
161  C228 10BF CCC1    LC228   STY   PINIT+1   BUILD A JUMP TABLE
162  C22C 31   23              LEAY  3,Y       POINTING TO THE RELOCATED
163  C22E 10BF CCD1            STY   $CCD1     PRINTER DRIVER
164  C232 31   23              LEAY  3,Y
165  C234 10BF CCE5            STY   POUT+1
166  C238 31   23              LEAY  3,Y
167  C23A 10BF CC09            STY   PCHK+1
168  C23E 31   23              LEAY  3,Y
169  C240 86   7E              LDA   #$7E      JMP INSTRUCTION
170  C242 B7   CCC0            STA   PINIT
171  C245 B7   CCD0            STA   PTERM
172  C248 B7   CCE4            STA   POUT
173  C24B B7   CC08            STA   PCHK
174  C24E 39                   RTS             RETURN
175  C24F 30   8C 1C   WMSG1   LEAX  <ERR01,PCR SPOOLING INACTIVE MESSAGE
176  C252 20   0E              BRA   LC262
177  C254 30   8D 00A7 WMSG2   LEAX  POPEN,PCR NO PRINTER DRIVER FROM P.COR
178  C258 20   08              BRA   LC262
179  C25A 30   8C 25   WMSG3   LEAX  <ERR02,PCR INVALID PORT NUMBER
180  C25D 20   03              BRA   LC262
181  C25F 30   8C 38   WMSG4   LEAX  <ERR04,PCR NOT ENOUGH USER MEMORY
182  C262 BD   CD1E    LC262   JSR   PSTRNG
183  C265 EC   8D 0080         LDD   FLIEND,PCR RESTORE MEMEND
184  C269 FD   CC2B            STD   MEMEND
185  C26C 20   70              BRA   PEXIT
186  C26E 20 20 20 53  ERR01   FCC   "-- Spooling Active.",EOT
     C272 70 6F 6F 6C
     C276 69 6E 67 20
     C27A 41 63 74 69
     C27E 76 65 2E 04
187  C282 20 20 20 49  ERR02   FCC   "-- Invalid Port Number.",EOT
     C286 6E 76 61 6C
     C28A 69 64 20 50
     C28E 6F 72 74 20
     C292 4E 75 6D 62
     C296 65 72 2E 04
188  C29A 20 20 20 4E  ERR04   FCC   "-- Not Enough User Memory."
     C29E 6F 74 20 45
     C2A2 6E 6F 75 67
     C2A6 68 20 55 73
     C2AA 65 72 20 4D
     C2AE 65 6D 6F 72
     C2B2 79 2E
189  C2B4 04                   LC2B4   FCB   $04
190  C2B5 7F   CC09    LC2B5   CLR   TTYPS
191  C2B8 BD   CCC0            JSR   PINIT     INITIALIZE PRINTER
192  C2BB BD   CD4B            JSR   GOCMND    EXECUTE COMMAND LINE
193  C2BE BD   CC80            JSR   PTERM
194  C2C1 30   8C F0           LEAX  <LC2B4,PCR
195  C2C4 BC   CC2B            CMPX  MEMEND
196  C2C7 26   06              BNE   LC2D2
197  C2C9 EC   8C 2A           LDD   <FLIEND,PCR RESTORE MEMEND
198  C2CC FD   CC2B            STD   MEMEND
199  C2CF EC   8C 22   LC2D2   LDD   <ATEMP,PCR RESTORE FLEX CONSTANTS
200  C2D2 B7   CC09            STA   TTYPS
201  C2D5 F7   CC2B            STB   CMDFLG
202  C2D8 EC   8C 10           LDD   <BTEMP,PCR
203  C2DB FD   CC43            STD   $CC43
204  C2DE 5F                   PEXIT CLRB            RESET PRINTER
205  C2DF 4F                         CLRA            PORT ADDRESS
206  C2E0 FD   CC39            STD   PRTDVC
207  C2E3 86   39              LDA   #$39      RETURN INSTRUCTION
208  C2E5 B7   CCC0            STA   PINIT
209  C2E8 B7   CCD0            STA   PTERM
210  C2EB B7   CCE4            STA   POUT
211  C2EE B7   CC08            STA   PCHK
212  C2F1 7E   C303            JMP   WARMS
213  C2F4                      ATEMP RMB   2         TEMPORARY STORAGE
214  C2F6                      FLIEND RMB  2         MEMEND
215  C2F8                      BTEMP RMB   2         TEMPORARY STORAGE
216  C300                      ORG   $C300
217                            *
218                            * THE FOLLOWING PARALLEL PRINTER DRIVER IS
219                            * WRITTEN IN POSITION INDEPENDENT CODE FOR
220                            * THE EPSON MX-80 PRINTER CONNECTED TO PORT 2B.
221                            *
222                            * This program accepts and processes printer control
223                            * codes of the form "^D" which are imbedded in the text
224                            * file being printed.  "D" can have the numeric values
225                            * 0 thru 9 and the alphabetic values "A" thru "J".
226                            *
227                            * This driver tests for a valid control code
228                            * and then sends a one or two byte sequence
229                            * defined in the PCTBL (printer control table)
230                            * to the EPSON printer to perform the appropriate
231                            * action.
232                            *
233                            * Invalid and out of range codes are printed as is.
234                            *
235                            ********************************************
236                            * SYSTEM EQUATE
237                 C000 FLEX  EQU   $C000
238                 CD36 ADDBX EQU   FLEX+$0036
239                            * EQUATES
240                            *
241                            *** PRINT ROUTINE
242                            *
243  C300 0154                 LENGTH FDB   EMOS-POPEN LENGTH OF DRIVER
244                            *
245                            *  ENTRY VECTORS
246                            *
247  >C302 16  0012    POPEN  LBRA  OPEN      PRINTER INITIALIZE
248  >C305 16  004B    PQUIT  LBRA  CLOSE     PRINTER TERMINATE
249  >C308 16  004E    PCHAR  LBRA  PUT       PRINT CHARACTER
250   C30B 16  0087    PCHEK  LBRA  CHECK     PRINTER READY CHECK
251                            *
252                            *   PARALLEL PRINTER FILE CONTROL
253                            *
254  C30E E070         PIA    FDB   $E070     DEFAULT PORT ADDRESS
255  C310 01           SIDE   FCB   $01       INTERFACE SIDE (1)
256  C311 00                  FCB   0         -- RESERVED BYTE --
257  C312 FF           PFLAG  FCB   $FF       PRINTER READY FLAG
258                 0000 DR     EQU   0        DATA REGISTER OF PIA
259                 0000 DDR    EQU   0        DATA DIRECTION REG OF PIA
260  C313 3C           PAGLN  FCB   60        LINES PER PAGE
261  C314 00           LINCT  FCB   0         LINE COUNT
262  C315 00           CAFLAG FCB   0         1 = PREV. CHAR WAS CR
263  C316 00           CCFLAG FCB   0         1 = PREV. CHAR UP ARROW
264                            *
265                            *   PRINTER INITIALIATION
266                            *
267  C317 34   16      OPEN   PSHS  A,B,X     SAVE REGISTER'S
268  C319 A6   8C F4          LDA   SIDE,PCR  TEST SIDE SELECT
269  C31C 2A   0C             BPL   ASIDE     DEFAULT VALUE
270  C31E 84   01             ANDA  #1
271  C320 48                  ASLA            MULTIPLY BY TWO
272  C321 A8   8C EB          ADDA  PIA+1,PCR FORM NEW PIA ADDRESS
273  C324 A7   8C E8          STA   PIA+1,PCR
274  C327 6F   8C E6          CLR   SIDE,PCR  NOW CLEAR SIDE DESIGNATOR
275  C32A B6   CC33    ASIDE  LDA   CPUTYP    TEST CPU TYPE
276  C32D 85   04             BITA  #4        DETERMINE INTERFACE TYPE
277  C32F 26   0D             BNE   S09
278  C331 EC   8C DA          LDD   PIA,PCR
279  C334 CA   0F             ORB   #$0F
280  C336 2B   06             BMI   S09
281  C338 1F   01             TFR   D,X       MOVE PORT ADDRESS INTO X REG
282  C33A B6   0F             LDA   #$0F      INITIALIZE LATCH
283  C33C A7   84             STA   0,X
284  C33E AE   8C CD   S09    LDX   PIA,PCR   RESTORE PRINTER PORT ADDRESS
285  C341 B6   3A             LDA   #$3A      SELECT DATA DIRECTION REGISTER
286  C343 A7   01             STA   1,X       BY WRITING INTO CONTROL REGISTER
287  C345 B6   FF             LDA   #$FF      SELECT ALL OUTPUT LINES
288  C347 A7   84             STA   0,X       PUT INTO DATA DIRECTION REGISTER
289  C349 B6   3E             LDA   #$3E      SET UP FOR TRANSITION CHECKS
290  C34B A7   01             STA   1,X       AND ENABLE OUTPUT REGISTER
291  C34D 68   84             TST   0,X
292  C34F 86   11             LDA   #$11      ION
293  C351 20   27             BRA   READY     OUTPUT CHARACTER
294                            *
295                            *   TERMINATE PRINTER PROCESSING
296                            *
297  C353 86   0D      CLOSE  LDA   #$0D      PUT CARRIAGE RETURN
298  C355 8D   02             BSR   PUT
299  C357 86   13             LDA   #$13      IOFF
300                            *
301                            *   PRINTER OUTPUT CHARACTER ROUTINE
302                            *
303  C359 34   16      PUT    PSHS  A,B,X     SAVE REGISTERS
304  C35B 8D   38      WAIT1  BSR   CHECK     TEST FOR PRINTER READY
305  C35D 2A   FC             BPL   WAIT1     LOOP UNTIL PRINTER IS READY
```

```
306  C35F AE  BC AC          LDI   PIA,PCR  GET PRINTER ADDRESS
307  C362 60  BC 01          TST   CCFLAG,PCR PREV. CONTROL CODE?
308  C365 27  03             BEQ   TESTCR   NO
309  >C367 16  006F          LBRA  PRCODE   YES
310  C36A 81  0D     TESTCR  CMPA  #$0D     CR
311  C36C 27  58             BEQ   CR       YES
312  C36E 81  0A             CMPA  #$0A     LINE FEED
313  C370 27  48             BEQ   LFD      YES
314  C372 81  5E             CMPA  #$5E     UP ARROW
315  C374 27  5F             BEQ   PCODE    YES - HANDLE IT
316  C376 81  0C             CMPA  #$0C     FORM FEED
317  C378 27  33             BEQ   FORM     YES
318  C37A 6F  BC 95  READY   CLR   PFLAG,PCR SET PRINTER FLAG NOT READY
319  C37D A7  84             STA   DR,X     SET DATA IN OUTPUT REGISTER
320  C37F 86  36             LDA   #$36     SET DATA READY, HIGH TO LOW
321  C381 A7  01             STA   1,X      STORE INTO CONTROL REGISTER
322  C383 86  3E             LDA   #$3E     THEN SEARCH FOR TRANSITION
323  C385 A7  01             STA   1,X      OF LOW LEVEL TO HIGH
324  C387 35  96             PULS  A,B,X,PC RESTORE REGISTERS
325  C389 34  16     PUTCHR  PSHS  A,B,X    SAVE REGISTERS
326  C38B 8D  08     WAIT2   BSR   CHECK    WAIT UNTIL PRINTER IS READY
327  C38D 2A  FC             BPL   WAIT2
328  C38F AE  8D FF7B        LDX   PIA,PCR  GET PRINTER ADDRESS
329  C393 20  E5             BRA   READY    PRINTER IS READY
330                  *
331                  *  CHECK FOR PRINTER READY
332                  *
333  C395 34  10     CHECK   PSHS  X        SAVE INDEX REGISTER
334  C397 6D  8D FF77        TST   PFLAG,PCR CHECK READY FLAG
335  C39B 2B  0E             BMI   CHEXIT   IF NEGATIVE, PRINTER READY
336  C39B AE  8D FF6D        LDX   PIA,PCR  PICK UP INTERFACE ADDRESS
337  C3A1 6D  01             TST   1,X      CHECK FOR TRANSITION
338  C3A3 2A  06             BPL   CHEXIT   IF PLUS, PRINTER NOT READY
339  C3A5 6D  84             TST   DR,X     RESET TRANSITION STATUS
340  C3A7 63  8D FF67        COM   PFLAG,PCR SET PRINTER READY FLAG
341  C3AB 35  90     CHEXIT  PULS  X,PC     RESTORE REGISTERS
342                  *
343                  ** HANDLE CR, LF, FF, PRINT CODES
344  C3A0 6F  8D FF63 FORM    CLR   LINCT,PCR CLEAR LINE COUNT
345  C3B1 8D  06             BSR   PUTCHR   PRINT FORM FEED
346  C3B3 35  96             PULS  A,B,X,PC
347                  * SET CONTROL CODE FLAG
348  C3B5 86  FF     PCODE   LDAA  #$FF     SET CONTROL CODE FLAG
349  C3B7 A7  8D FF5B        STAA  CCFLAG,PCR
350  C3BB 35  96             PULS  A,B,X,PC
351                  * HANDLE LINE FEED
352  C3BB 6D  8D FF54 LFD     TST   CRFLAG,PCR LF FOLLOWING CR
353  C3C1 27  10             BEQ   LFD1     NO MIDLINE
354  C3C3 6F  8D FF4E        CLR   CRFLAG,PCR YES CLEAR FLAG
355  C3C7 35  96             PULS  A,B,X,PC
356                  * HANDLE CARRIAGE RETURN
357  C3C9 8D  0E     CR      BSR   PUTCHR   PRINT CR
358  C3CB 86  FF             LDAA  #$FF     SET CR FLAG
359  C3CD A7  8D FF44        STAA  CRFLAG,PCR
360  C3D1 86  0A             LDA   #$0A     OUTPUT A LINE FEED
361  C3D3 8D  04     LFD1    BSR   PUTCHR
362  C3D5 8D  30             BSR   PAGE     PAGE END?
363  C3D7 35  96             PULS  A,B,X,PC RESTORE REGISTERS
364                  * PROCESS PRINTER CONTROL CODES
365  C3D9 81  30     PRCODE  CMPA  #'0      RANGE CHECK 0 - 9
366  C3DB 25  2E             BLO   PRINTC   LT "0" - PRINT CHAR
367  C3DD 81  39             CMPA  #'9
368  C3DF 2F  0A             BLE   NUMER    NUMERIC VALUE
369  C3E1 81  41             CMPA  #'A      IS CHARACTER LT "A"
370  C3E3 25  26             BLO   PRINTC   YES - PRINT CHAR
371  C3E5 81  4A             CMPA  #'J      IS CHARACTER GT "J"
372  C3E7 22  22             BHI   PRINTC   YES - PRINT CHAR
373  C3E9 80  07             SUBA  #7       SKIP 7 CHARACTERS
374  C3EB 80  30     NUMER   SUBA  #'0      CONVERT TO BINARY
375  C3ED 81  0C             CMPA  #12      FF FLAG
376  C3EF 26  04             BNE   TABLE
377  C3F1 6F  8D FF1F        CLR   LINCT,PCR YES CLEAR LINE COUNT
378  C3F5 48         TABLE   ASLA           MULTIPLY BY TWO
379  C3F6 1F  894B           TAB            MOVE INTO D
380  C3F9 30  8D 0031        LEAX  PCTBL,PCR POINT AT CODE TABLE
381  C3FD 8D  D836           JSR   ADDDX    GET ADDRESS OF PTR CODE
382  C400 A6  84             LDAA  0,X      GET FIRST CONTROL CODE
383  C402 27  0A             BEQ   NOCODE   ZERO - NOT VALID FOR EPSON
384  >C404 17  FFB2          LBSR  PUTCHR   OUTPUT CHARACTER
385  C407 A6  01             LDA   1,X      GET SECOND BYTEA
386  C409 27  03             BEQ   NOCODE   ZERO - NOT VALID FOR EPSON
387  C40B 17  FF7B   PRINTC  LBSR  PUTCHR   OUTPUT CHARACTER
388  C40E 6F  8D FF04 NOCODE CLR   CCFLAG,PCR CLEAR CODE FLAG
389  C412 35  96             PULS  A,B,X,PC RESTORE REGISTERS & RETURN
390                  * CHECK PAGE LENGTH
391  C414 A6  8B FEF8 PAGE   LDAA  PAGLN,PCR LOAD PAGE LENGTH
392  C418 27  13             BEQ   PAGE1    IF ZERO, NO CHECK
393  C41A 6C  8D FEF6        INC   LINCT,PCR INCREMENT LINE COUNT
394  C41E A1  8D FEF2        CMPA  LINCT,PCR PAGE DONE
395  C422 24  09             BHS   PAGE1    NO - SKIP
396  C424 6F  8D FEEC        CLR   LINCT,PCR YES - CLEAR LINE COUNT
397  C428 86  0C             LDAA  #$0C     SEND FORM FEED
398  C42A 17  FF2C           LBSR  PUT      OUTPUT IT
399  C42D 39                 RTS
400                  * PRINTER CODE TABLE
401                  * DESIGNED FOR AN EPSON QX F/T
402                  *
403  C42E 7F00       PCTBL   FDB   $7F00     0  PRINTER BACKSPACE
404  C430 0000               FDB   $0000     1  START UNDERLINE
405  C432 0000               FDB   $0000     2  END UNDERLINE
406  C434 1B45               FDB   $1B45     3  START EMPHASIZED
407  C436 1B46               FDB   $1B46     4  END EMPHASIZED
408  C43B 0E00               FDB   $0E00     5  START DBL. WIDTH
409  C43A 1400               FDB   $1400     6  END DBL. WIDTH
410  C43C 0700               FDB   $0700     7  BELL
411  C43E 0000               FDB   $0000     8  NOT USED
412  C440 0000               FDB   $0000     9  NOT USED
413  C442 0A00               FDB   $0A00     A  LINE FEED
414  C444 1B3C               FDB   $1B3C     B  HOME HEAD
415  C446 0C00               FDB   $0C00     C  FORM FEED
416  C448 0D00               FDB   $0D00     D  CARRIAGE RETURN
417  C44A 1B14               FDB   $1B14     E  START ITALICS
418  C44C 1B35               FDB   $1B35     F  END ITALICS
419  C44E 1B47               FDB   $1B47     G  START DOUBLE STRIKE
420  C450 1B48               FDB   $1B48     H  END DOUBLE STRIKE
421  C452 0F00               FDB   $0F00     I  START COMPRESSED
422  C454 1200               FDB   $1200     J  END COMPRESSED
423                  C456 ENDS  EQU   *
424  CCC0                    ORG   $CCC0
425  CCC0 7E  C1B2   PINIT   JMP   LC1B2
426                          END   PCMB

0 ERROR(S) DETECTED

SYMBOL TABLE:

ADDBX  C836  ASIDE  C32A  ATEMP  C2F7  BTEMP  C2FB  CCFLAG C316
CHECK  C395  CHEXIT C3AB  CLOSE  C353  CMDFLG CC2B  CPUTYP CC33
CR     C3C9  CRFLAG C315  DDR    0000  DDCMND C04B  DR     0000
DUMMY  B3E0  ENDS   C456  ENTRY  C107  EOT    0004  ERR01  C271
ER903  C285  ERR04  C290  FLEI   C000  FLIEND C2F9  FORM   C3AB
IN9EC  C04B  LC147  C147  LC14E  C14E  LC1A0  C160  LC170  C170
LC17B  C178  LC196  C196  LC1B2  C1B2  LC1CC  C1CC  LC1D3  C1D3
LC1DE  C1DE  LC1E5  C1E5  LC1F8  C1F8  LC201  C201  LC202  C202
LC20A  C20A  LC21C  C21C  LC21E  C21E  LC228  C228  LC265  C265
LC2B7  C2B7  LC2B8  C2B8  LC2D2  C2D2  LENGTH C300  LFD    C3BB
LFD1   C3D3  LINCT  C314  LSTTRM CC11  MEMEND CC2B  NOCODE C40E
NUMER  C3EB  NTICH  C027  OPEN   C317  OUTCH  C00F  PAGE   C414
PAGE1  C42D  PAGLN  C313  PCHAR  C30B  PCHEK  C30B  PCMB   CCD0
PCMB   C100  PCODE  C3B5  PCTBL  C42E  PEXIT  C2E1  PFLAG  C312
PIA    C30E  PINIT  CCC0  POPEN  C302  POUT   CCE4  PRBIT  C305
PRCFLG CCFC  PRCODE C3D9  PRINIC C40B  PRTADR CC35  PRTDVC CC39
PRTLNG CC37  PSTRNG CD1E  PTERM  CCD0  PUT    C359  PUTCHR C389
READY  C37A  S09    C33E  SIDE   C310  TABLE  C3F5  TESTCR C36A
TTYPS  CC09  T_INIT D3F1  T_OFF  D3ED  T_ON   D3EF  WAIT1  C38B
WAIT2  C38D  WARMS  CD03  WMSG1  C252  WMSG2  C257  WMSG3  C25D
WMSG4  C262
```

60 Micro Journal
5908 Cassandra Smith
PO Box 849
Nixon, Tenn. 37343

23 January 1984

Dear Don and Readers,

I'd like to take this op rtunity, if I ay, to mark the passing of my colleague, mentor, and best friend Bill Henry. Bill, who died on December 1, 1983, was one of the 68 Micro Journal's biggest fans. He had been a syste a analyst for many years, and was never very interested in micros until we both discovered the 6809 (and the Journal) a few years ago. Since then, he could speak of nothing else. He just thought the Journal was t e best thing since I don't know what (especially the editorial policy), and as an experianced and thoughtful programmer he thought the '89 was about the most right-headed archetecture he'd ever seen

(except, maybe, the Burroughs B-6700). In fact, when he began to teach me about writing compilers for structured languages such as Algol and Pascal, he used the '09 as a model of a near-perfect stack machine (that is, until the 68000 came out).

Even though Bill lived most of his 56 years in Greenwich Village here in New York, I think he felt more in common with the far flung readers of the Journal than he did with many of his "intellectual" colleagues. He always said that the Journal was evidence that there is still hope for the world, because there are so many obviously sane and rational people "out there" in places like, for instance, Hixon Tennessee, using what anyone with half a brain can see is the ONLY way to write efficient, fun, and structually and morally "right" code, namely the 6809!

Bill was an extraordinary man. All of us who knew him feel a sense of loss which is impossible to describe. I want to thank the Journal, and it's readers, for giving Bill such a good time.

Sincerely,

Dell W. Setzer
254 West 88 St.
New York, NY 10024

Ed's Note: Dell, we here at 68 Micro Journal join you in marking the passing of Bill Henry. As a man far wiser than I once wrote - "With the death of any man, a little of all of us also dies". I am sure that the 68XX Community will be a little poorer for his passing.
DMW

Dear Mr Williams,

I noticed in a recent article that there was a reference to a 68 Micro Journal bulletin board. I now have a modem and a communications package and would like to begin using it. Could you please send me the telephone number of your bulletin board. Also, do you have the numbers of any other bulletin boards for 68XX users.

Dan Eisenberg
2288 Society Hill
Cherry Hill, NJ 08003
615-842-6809

Ed's Note: Dan, The phone # is (615)842-6809 for our bulletin board. Also there are several other bulletin board files, on the 68 Micro Journal BB, use 6809 number above. Most end with .BBS. Have fun and leave something useful or handy for the rest of us.

OS-9 USERS GROUP NEWS LETTER

President's Column
By Dale L. Puckett

The OS-9 User's Group has been moving right along since that fateful day in mid-August when several hundred of you -- gathered in Des Moines, IA for the second annual Microware OS-9 seminar -- elected new officers and gave us a mandate to make it go. First, let me introduce your new officers.

Peter Dibble, our Vice President, is the Head of User Services at the University of Rochester in New York. An avid OS-9 fan, Pete writes the monthly OS-9 column for **68 Micro Journal**. He knows OS-9 backwards and forwards and has even written an alternate Shell which you may see in our User Group Software Exchange Library some day soon.

Tom Murphy, our Secretary, is President of Suntel Systems Corp. in Earth City, Mo. In case you're wondering, Earth City is a suburb of St. Louis. Tom uses OS-9 daily in his business which serves telephone companies around the country. His GIMIX hardware and OS-9 software join forces to monitor operator-handled traffic in the nationwide telephone network. Suntel helps its customers get the most efficient use out of their equipment.

George Dorner, our treasurer, is an educator and veteran OS-9 User. He teaches at Harper College, a community college in Rolling Meadows, a Chicago suburb. George is one of the few who run OS-9 on an Apple. He uses the "Mill" and served our User's Group last year by running the Bulletin Board from a timesharing computer at the college.

Your President is a warrant officer in the U. S. Coast Guard and author of Microware's new book, "A Complete Tour Guide of BASIC09". He also writes a monthly OS-9 column for **Rainbow**, a magazine dedicated to Color Computer users. He uses a GIMIX Level II system and is the author of Eather, Readtest, Help and DynaSpell on 6809 based computers and co-author of "The Speller" from the Hayden Software Company on the IBM PC. An amateur radio operator since 1957, he has been using 68XX microcomputers since 1975 when he cut his teeth on a SWTPC rig with MIKBUG and 12K of memory.

KUDOS BRIAN

Thanks to Brian Capouch and his officers for coming up with the idea of an OS-9 Users Group and holding it together during that first trying year. Brian pushed the idea of a Software Exchange for OS-9 Users and gave it the boost needed for a quick start at the OS-9 Seminar this past August. You'll see his name several times in the list of software available in our library.

MORE KUDOS

Thanks to Ken Kaplan and all the programmers at Microware for the encouragement they gave us.

And, thanks to Jeanne Kaplan and the other gals at Microware who signed us up with our own Post Office box and promised to faithfully forward our mail each week.

Thanks to Richard Don at GIMIX who donated a Level III system for our bulletin board. And, to Bobby Phillips for letting him do it and Mike Magnus for maintaining it. You'll read a lot about our new bulletin board in George Dorner's column.

Thanks to Frank Hogg at FHL who has offered to duplicate and distribute our Users Group software.

Thanks to Jim Bellomo who set up an OS-9 Users SIG on Compuserve. His user ID is 71625, 240 if you want to contact him.

Thanks to James Widdowson at KEMTRONIX (UK) LTD, High Street, Compton, Berkshire, England RG16 ONL for donating a copy of DynaMail to the Users Group officers so that we can maintain our membership list easily. DynaMail is a selective indexed mailing list and marketing analysis system.

Thanks to Dave Lissiuk, Software Manager at Springbok Digitronics in Silver Springs, MD and Milan Chris Getting of the J*% Group in King of Prussia, PA for volunteering to assist with the communicationa committee.

Thanks to Don Williams at 68 Micro Journal and Color Micro Journal, Bill Sias at Forum Sixty-Eight, Dr. Bud Pass at System 68 and Lonnie Falk at Kaiobow who agreed to publish all the information we could write about the users group.

And finally, thanks to Dr. Rudolf Keil of Germany who volunteered to be our contact point in Europe. He is maintaining contact with George so that we have a way to get information to and from Europe.

OUR CHAIRMEN

Two members deserve special credit and recognition. Dave Kaleita, manager of the engineering and test department at the Jabil Circuit Co. in Troy, Michigan volunteered to chair our Software Exchange Committee. Dave is an active member of the Southeast Michigan Computer Club. He's doing a great job despite the fact that this is probably the hardest and most time consuming job in the Users Group.

Tom Westhoff of Westronics in Willmar, MN is the chairman of our communications committee. He is designing a bulletin board that will look and act like OS-9. In fact, it will be OS-9. We feel that operation will be a snap for the oldtimers and a tremendous way to introduce OS-9 to new users. You'll read more in George's column.

OUR PURPOSE

We'll start by quoting from our new by-laws.

"The purpose of the OS-9 Users Group is:

1. To stimulate and sustain interest in computers in general, and in OS-9 in particular.

2. To promote the cooperation and exchange of information between members.

3. To conduct programs and activities to promote fraternalism and to advance the general interest and knowledge of members.

When you elected us in August, you gave us three mandates:

1. Incorporate the Users Group and if possible obtain a tax exempt status.

2. Establish a method of communication to and from our members.

3. Coordinate with overseas OS-9 users and attempt to establish a continuous flow of information across both the Atlantic and Pacific.

After the election, your officers met and established some secondary goals to augment the three tasks you mandated. We decided that building our membership would be one of our foremost goals. By doing this, we would create the opportunity for local clubs to form in many cities. Through this social medium our new members would have the chance to learn from the oldtimers. The local clubs could then communicate their desires through our bulletin board and Compuserve.

We also wanted to emphasize education. We thought that by getting the oldtimers to tell sea stories to new OS-9 users in local clubs we could serve the many newcomers brought to us by Tandy's introduction of OS-9 on the Color Computer.

Speaking of education, here's a great idea I picked up from Richard Don at Gimix. We could put a CoCo "Help" directory on our OS-9 bulletin board. This directory could be packed full of hints for the new OS-9 user and answer those questions we all had when we first started using this outstanding operating system. Do we have any volunteers? Call George Dorner if you would like to help work something like this up. When your pride and joy is on the air, Pete and I will spread the word in the trade press.

We hope to coordinate your wishes and make this Users Group a viable voice for the OS-9 User so we will be in a position to make your desires known to the management at Microware. We will tell them which utilities you would like to see developed and the type of enhancement you would like to see in the future. We will strive to bring solid support to every OS-9 user.

We knew that if we were to achieve the goals stated above we would need a large active membership. To get that we would need an incentive. We decided that the most effective incentive would be to establish a Software Exchange which gave you access to a large selection of OS-9 software.

In an attempt to accomplish all of this we decided to use a structured approach. Four committees would form the heart of the Users Group. Each officer would work with the chairman of one of the committees.

Tom Murphy, our secretary tackled the administrative chores with the help Tom Westhoff, George Dorner, Brian Capouch and others.

George Dorner heads up the communications effort. He works hand in hand with Tom Westhoff, chairman of our Communications committee and Dr. Keil our point of contact with European OS-9 users.

Peter Dibble tackled the membership problem and is moving the effort along nicely with the assistance of Ken Kaplan and the staffers at Microware, Richard Don at GIMIX, Frank Hogg at FHL and many others in the industry.

And finally, I agreed to work with Dave Kaleita to help establish a viable Software Exchange program. Dave has done a tremendous job and you'll see the impressive catalog he has mustered elsewhere in this newsletter. Thanks to Frank Hogg's offer to take care of our duplication and distribution we are going to be able to deliver user written, public domain software to you for only $3.00 per disk. Further, we hope to pack at least 10 programs on each disk.

As a bare minimum, everyone who joins the OS-9 Users Group will receive one disk free. Dave will select programs that he thinks are needed in everyone's "toolbox." Included will be a simple data entry program that will make it easy for you to submit your own programs and a modem program so you can get on line and talk to our OS-9 Users Group bulletin board.

So, with this resolve and much enthusiasm we left Des Moines and began our planning. Our next meeting would be a conference telephone call in early September. That meeting went well, as did the second conference call in mid-October. You'll see the results of those calls in each officers committee report in this newsletter.

POST OFFICE BOX

Thanks to Ken Kaplan and the office staff at Microware, we now have our own mailing address. Please send all mail to us at this address. Ken's secretaries say they will empty the mailbox once a week and mail it to one of the officers. He in turn will see that the proper officer or committee chairman answers your correspondence. Here's the box number:

OS-9 Users Group
P. O. Box 8027
Des Moines, IA 50301

SOFTWARE EXCHANGE NEWS
By Dave Kaleita

We've accomplished a lot since we left Des Moines last August. Thanks to quick decisions from our new officers and a lot of friends in the industry we will soon have a very exciting software exchange program. In fact not too long after you read this newsletter, you should be receiving your first disk.

The first program on our first disk is a quick data entry utility that will make your future submissions to our library quite painless. You will only need to answer a few questions, copy your program on to the disk and put it in the mail.

Here's how our software exchange will work. First, all current members will receive one disk containing a selection of programs. In addition to the quick data entry program mentioned above, we also hope to give you a modem program and eight to 10 additional utilities. Each new member will also receive this disk. When Pete receives a membership application and the annual membership fee ($25.00), he will immediately forward a mailing label to Frank Hogg who will ship the disk.

I'll be putting together several additional disks, each containing eight to 10 programs. They'll probably be grouped by type, i.e., BASIC09, 'C', PASCAL, COBOL, etc. As I complete the new disks, I'll forward the master to Frank, who will keep it on file. We'll publish a list of the utilities and programs available on these disks in the trade pubs.

There are two ways for you to get one of these additional disks. If you have a program you would like to donate to the Users Group, send it to us at Post Office Box 8027, Des Moines, IA 50301. You will receive your choice of one of the additional disks FREE.

If you don't have a program to donate, don't worry. It's still almost painless. To receive an additional disk, send us a letter with the number of the disk you would like and $3.00 dollars. This will cover the expense of making the disk. You may buy as many of the additional disks as you like. They will all be available for $3.00 each.

Please remember however, if this software exchange is going to work, we are going to have a continuous input of software. That means somewhere along the line you are going to have to contribute your share.

We'll all be ahead in the long run. It makes absolutely no sense for each of us to spend all that time re-inventing the wheel. The OS-9 Users Group Software Exchange is an exciting answer to that problem. We all stand to gain. All software will be Copyrighted in the name of the original author and the OS-9 Users Group.

We will be able to ship these disks in five-inch, eight-inch standard OS-9 and five-inch Color Computer OS-9 formats. Please let us know which type you need when you order.

BULLETIN BOARD NEWS
by George Dorner

Plans for a real OS-9 Bulletin Board go forward ploddingly. Hardware is not the problem. GIMIX has donated space on a GIMIX III at their offices in Chicago and the hardware baby-sitting will be volunteered by Mike Magnus. Thanks to Richard Don for making these arrangements. The officers have voted to acquire a 300/1200 baud modem for use on the BBS, if other arrangements fall through. A phone line has been ordered for installation at GIMIX which the Users Group will pay for.

Per usual, the software is the real slowdown. We do have Dave Lissiuk's large program, but it hasn't been fully tested yet. At Des Moines and in subsequent discussions, we decided to implement a "bare bones" system at first. This will likely be menu driven, but will have the option to exit that structure so the user may just use his knowledge of OS-9 to use the system. Certain functions of OS-9 will be defeated to protect the system. This is well along the way I am told, but I haven't seen the software at this point. If it isn't forthcoming, there are some other options to produce BBS software - but at the cost of more time.

When the OS-9-based OS-9 BBS comes to life, it will be announced on the old OS-9 BBS, on the COCO OS-9 section of CompuServe, and on the CompuServe OS-9 SIG (assuming that it exists by that time). If you would like to help out on getting the BBS off the ground, let me know. I have a couple of volunteers and would like to establish a broad-based committee to bird-dog this issue.

OS-NINE, the First OS-9 BBS
George Dorner

At the May 1982 Microware meeting in Des Moines, we offered to set up a temporary bulletin board system until a "real" OS-9 BBS could be launched. Alas, a year and a half later this BBS, known as OS-NINE to me and no one else, still exists and gobbles up several hours a month of my free time. It will continue to exist until it is no longer needed. I hope to see it put to death by the imminent birth of the "real" system as noted elsewhere in this newsletter.

OS-NINE functions on an HP Access (or HP 2000G) minicomputer. This was a super computer for timesharing about ten years ago when it was young, but it has some real disadvantages for a BBS system. Chief among these is the inability to

interrupt output from the keyboard, other than with a break key which serves as an "exit" command. Thus, it is hard to browse without building up a large phone bill. A listing may only be aborted with a 'break' and that will be after a 128 character buffer empties!

The other shortcoming is that the system uses a bastardized X-on/X-off protocol which sends many [control S] (hex $13) characters within any message. This is OK for a dumb terminal, but it will confuse most OS-9 terminal programs unless these characters are stripped out.

Despite these two bad features and the fact that the software has purposely NOT been made more friendly or otherwise improved, lots of useful and interesting information has passed there. Most of the "big names" of the OS-9 world have been there at least once, and a query almost always is answered within a week.

How to Log On OS-NINE

***Dial 312-397-8308 or 312-397-8380 (Chicago area)

***When connected, type [CR][LF] or CTRL-M CTRL-J several times until you see:

PLEASE LOG IN

***then you should type:

bel-g500,,3 and a [CR].

***This should get you in and you should see:

```
User:  4729      Last:  562
)     OS-NINE users group BBS


Use [?] to see commands.

Archives in #1-#499.
Index in #10 ... needs some work.

Recent messages are after #500.

Users Group Stuff:  #501-510
CompuServe SIG:     #511-520
COCO/OS9:           #521-530


BREAK stops printing ... only after 128
char. buffer empties.

      You are User # 4729
      There are  244   Messages
      562   Is the Last Message Number


Function: E,G,H,K,Q,R,S,X
 (or ? if you don't understand)  q

Starting  Message # or -10 to start 10 back
from most recent :-5


Mess. Num.      Subject
 557          CoCo OS-9 Support ---K. Kaplan
 558          OS-9 Support --- S. Bassett
 559          To Kent Meyers
 560          COCO BBSRS
 561          Coco OS-9 --Ty Taylor
 562          68000 CROSS ASSEMBLER
Function: E,G,H,K,Q,R,S,X
 (or ? if you don't understand)   r 10

Mess # 10   Subject:  OS-NINE BBS Topical
Index         From:  Sysfolk

Dated:  DEC. 30,1982  # read 161
        Last Time Read  NOV. 5,1983


OS-NINE BBS Index Tree

Message No.     Topic Heading
1-99      General Communications
          1-9         Info About This BBS
     ***  10          OS-NINE BBS Index  Tree
(This message.)
          11-49       OS-9 Users' Group Info
                      11-29      General
                      31-39      Newsletter
                      41-49      Our Dream BBS
          51-59       Other B. B. Systems
          61-79       Miscellaneous
          81-89       Books and Publications
          91-99       Ham Radio/Packet Radio


101-299   Software Info
               101-149      General OS-9
               151-199      Problems and
                             Documentation
          201-270  Languages - General
               211-239      BASIC09
               241-249      Pascal
               261-269      C
          271-284  Utilities
          285-300  Modem/Communications
                   Software Info


301-349   Hardware Info
          301-309   6809/68000 Info
          311-319   Radio Shack COCO
          321-329   Gimix
          331-339   The Mill/Apple & 6809
          341-349   Others
351-399   Queries and Open Questions
401-499   Commercial Information
          401-449   Commercial Software
                    Info
          451-499   Commercial Hardware

That was message # 10


Message # to Retrieve or Return for next
 or 0 to quit :0

Function: E,G,H,K,Q,R,S,X
 (or ? if you don't understand)  g

Do you wish to leave a comment
 or helpful hint (Y/N)  n

Thanks for calling
0003 MINUTES OF TERMINAL TIME
```

Continued Next Month

CLASSIFIED ADVERTISING

LIKE NEW - ADDS-VIEWPOINT CRT TERMINALS - DEMO UNITS, THESE ARE LIKE NEW - WARRANTY NEVER REGISTERED - SAVE - ONLY **$478.00 EACH** - ONLY 1 LEFT. CALL ASK FOR DON, TOM or LARRY

A/C 800 338-680

•••

TELETYPE Model 43 PRINTER - with serial (RS232) interface, and full ASCII keyboard. **LIKE NEW** - New cost $1295.00 - ONLY **$759.00** ready to run - Call Tom - Larry - Bob, CPI 615 842-4600

•••

"FOR SALE - SWTPC CDS2 40MB DISC, 240V Power Supply, all Boards and Cables. Has had very little use. Has crashed head. Needs factory repair. Must sell $2,200.00 or best offer. Ring George Samids, Brisbane, Australia. 07-3979000 (work)"

•••

FOR SALE: SWTPC 69/A: MP-09, MP-8M, MP-S, MP-L2, DC-4, GIMIX 32K board w/20K installed, FLEX 9, ASM8, EDIT $750. Televideo 912C $400. Epson MX-80 with Graftrax $400. Percom LFD-400 w/PSYMON and Broken Siemans SS/SD drive $20.
Robert Alexander (617)872-6739 evenings.

•••

FOR SALE: F & D Disk Controller, single density, $75; SS-30 5" & 8" Disk Controller, single/double density, single/double sided with software, $175; Heathkit H-9 terminal, $80. Phil Moore, P O Box 2312, Florissant, MO 63032.

•••

FOR SALE - Terminus Arcade 50 Board with connector set and FBASIC $200.
Hazelwood VC-256 Graphic Board $200.
Call Tom 1-800-338-6800.

•••

COMPILER EVALUATION SERVICES
By: Ron Anderson

The S.E. MEDIA Division of Computer Publishing Inc.,
is offering the following 'SUBSCRIBER SERVICE':

COMPILER COMPARISION AND EVALUATION REPORT

Due to the constant and rapid updating and enhanchment of numerous compilers, and the different utility, appeal, speed, level of communication, memory usage, etc., of different compilers, the following services are now being offered with periodic updates.

This service, with updates, will allow you who are wary or confused by the various claims of compiler vendors, an opportunity to review comparisons, comments, benchmarks, etc., concerning the many different compilers on the market, for the 6809 microcomputer. Thus the savings could far offset the small cost of this service.

Many have purchased compilers and then discovered that the particular compiler purchased either is not the most efficient for their purposes or does not contain features necessary for their application. Thus the added expense of purchasing additional compiler(s) or not being able to fully utilize the advantages of high level language compilers becomes too expensive.

The following COMPILERS are reviewed initially, more will be reviewed, compared and benchmarked as they become available to the author:

PASCAL "C" GSPL WHIMISCAL PL/9

Initial Subscription - $39.95
(Includes 1 year updates)
Updates for 1 year - $14.50

S.E. MEDIA - CPI
5900 Cassandra Smith, POB 794
Hixson, TN 37343
615 842-4601

WORD PROCESSING SOFTWARE

SPELL 'N FIX finds and fixes your spelling and typographical errors. Cassette or disk versions cost just $69.29 with a 20,000 word dictionary. FLEX version $178.58
75,000 word optional dictionary costs $50 additional
ALL-IN-ONE editor word processor mailing list program costs $50 (STAR DOS version) or $75 (FLEX version)

DISK OPERATING SYSTEM

STAR-DOS provides the power of a big DOS with the simplicity of standard R S disk format. $49.90 for 16K 64K systems

SYSTEMS SOFTWARE

HUMBUG is the famous 6809 monitor debugger adapted to the CoCo. $39.95 for 16K or 32K disk or tape systems. $59.95 for 64K systems using STAR DOS or FLEX. $29.95 for the MC-10.
REMOTERM lets a terminal or modem control the CoCo or MC-10 for $19.95. Disk or cassette
COMMTERM communications terminal program for the MC-10 costs just $19.95
NEWTALK reads out memory contents in words through the TV speaker. $20. Disk or cassette

HOME FINANCE

CHECK 'N TAX lets you check on your bank. Not just a checkbook balancing program, but a help at tax time too. $50, disk only.

EDUCATION

Numerical Methods is a college level course on computer mathematics. $75, disk or cassette.

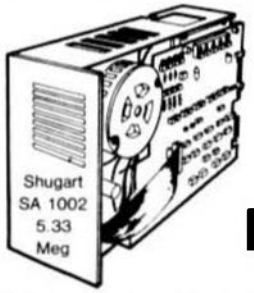
GAMES

SHRINK is our version of Eliza for $15. Disk or cassette.
THREE-D teaches spatial relationships through three-dimensional tic tac toe. $25, disk or cassette.

STAR-KITS

P.O. BOX 209 - M
MT. KISCO, N.Y. 10549
(914) 241-0287

HARD DISK DRIVE EXPANSION FOR SS-30 and SS-50 SYSTEMS



$269.00
5 Meg Hard Drive

This unbelievably low price applies to a limited quantity of new, unused drives in their original sealed shipping cartons, sold as-is — $269

Western Digital Controller Board (WD1001) will control up to four of these drives.
90-day warranty $331

DON'T MISS OUT - ORDER NOW!

DIGITAL SEARCH, 103 S. Main Street
Greer, SC 29651 803-877-9444

ASSEMBLERS

TSC

Macro Assembler
the FLEX STANDARD Assembler
F,CCF $50.00

Relocating Assembler w/Linking Loader
Use with many of the C and Pascal Compilers.
F,CCF $150.00

Great Plains Comp. Co.

RMAC
Relocating, Recursive-Macro Assembler
and Linking Loader for the 6809
Supports Recursive Macros, Conditional Assembly, etc. Optional X-Ref Listing. Includes a Small Line-oriented Editor as part of the Assembler.
F,CCF $150.00
F,CCF w/Source $200.00

OmegaSoft

SMALL
Relocating Assembler and **Linking Loader**
2-pass Relocating Assembler; 2-pass Linking Loader. Supports 6 Char. Labels, System Calls (SWI[x] FCB x), Expressions with Arith., Logic, and Shifts, etc.
F,CCF $125.00
One Year Maint. $50.00

Mindrush Micro Systems

EMACE
By Graham Trott. A combined Editor and Assembler designed to allow the Programmer to Enter, Edit, and Assemble Programs with a minimum of effort, w/o leaving the Program. Designed primarily for small to medium sized Program Development. Includes **EMACE**, a **Cross Assembler** for the MC6800/1/3 and Hitachi HD6301 (CMOS 6801).
F,CCF - $90.00

DISASSEMBLERS

Computer Systems Consultants

SUPER SLEUTH
Computer Systems Consultants **Super Sleuth** is a "Time Tested", reliable, PROVEN Disassembler that has gained acceptance through out the SS-50 Bus Community as an extremely POWERFUL, INTERACTIVE, Software Tool. The **Super Sleuth** Software Package consists of 3 Programs: **SLEUTH** (the **Disassembler**), **CHGREF** (used to globally Change labels to a meaningful Name), and **XREF** (a Cross Reference Generator for Source Code Files). SLEUTH will Disassemble Memory Resident 6809 Code and 6800, 6801, 6802, 6803 (the "Baby CoCo"), 6805, 6808, 6809, and 6502 (Apple, Atari, Commodore, etc.) Binary Disk Files. (See Aug. '83 '68' Micro Journal "Color Users Notes" Column for a full Review.)

Color Computer — **OS-9 Bus (all w/ Source)**

CCD (32K Req'd)
Obj. Only $49.00
CCF, Obj. Only $50.00
CCF, w/Source $99.00
CCD, Obj. Only $50.00

F, $99.00
U, $195.00
O, $101.00

ALL Computer Systems Consultants Software
runs on the **Color FLEX** Systems
ALL in stock
call 800-338-6800
for **DOORSTEP DELIVERY**

Computer Systems Center

CYMATE +
An "easy to use", powerful Disassembler for Disk Resident 6809 and 6800 Binary Files. Allows the development of a "Control File" of various Program "BOundaries" during successive disassemblies; can use a Label File which automatically replaces a Hex location with a Label Name; includes an XREF Utility; etc. Label Files provided for Mini-FLEX, FLEX2, FLEX9, Color Computer (for use with Color FLEX Systems), etc. OS-9 Version includes special OS-9 options.

CCF, Obj. Only $100.00
CCD, " " $150.00
F, " " $100.00
O, " " $150.00
U, " " $200.00

COMPILERS & DECOMPILERS

6809 "Structured" Assembly Lang. Compilers

Mindrush Micro Systems

PL/9
By Graham Trott. A "Structured" Assembly Language Editor/Compiler/Debugger, all in **ONE PACKAGE** provides a totally **INTERACTIVE** Program Development Cycle. The Compiler supports large Symbol Names, Variable Types, Pointers, Control Structures, Stack, A-, B-, and D-Register manipulation, etc. The **Source-Oriented** Trace/Debugger provides Single Stepping, Breakpointing, etc. An excellent Software Development Tool for utilizing the power of the 6809 in developing small to medium sized packages.
F, CCF - $198.00

Whimsical Developments

WHIMSICAL
Need the Ease of Design and Maintainability of "Structured Programming" **AND** the Speed and Control of Assembly Language? Then **WHIMSICAL** was designed for you! This Single Pass, Recursive Descent Compiler provides the tool for developing simple Utilities to MAJOR Systems in Assembly Language. Supports 3 "Lex" Levels which allow one level of Procedure nesting, or more within "Modules". It is easy to develop programs written for other machines since you are working at the Assembly Language level. Features unified, user-defined I/O; Produces ROMable, relocatable, recursive, re-entrant Code; Structured style and statements with Procedures and Modules; supports Byte and Double-Byte primitives with 3 types of Integers (up to 32 bit), Char and Boolean, and unlimited sized Arrays (vectors only); Interrupt handling; unlimited length Variable Names; Variable Initialization (defaults to $00); Include "Source File" directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. To quote Ron Anderson in his review of **WHIMSICAL** in the Sept. '83 Issue of **'68' Micro Journal** that, except for the lack of floats, "..., **I have to give this one VERY high rating.** ...". It is a FAST Compiler which produces FAST Code (his "Primes" Benchmark ran at 9 secs. on a 2 MHz System).
F and CCF - $195.00

'C' Compilers

Mindrush Micro Systems

C Compiler
By James McCosh. Full featured C Compiler for the FLEX Operating System. Includes a Reloc. Asmb., but needs the TSC Relocating Assembler/Linking Loader (which includes a Library Manager) for those "full blown" System Packages.
F and CCF - $295.00

Introl

C Compiler
A full-featured C, streamlined for the 6809. Generates very efficient object code. Output "benchmarks" close to 10MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz Z80 when using a 2MHz 6809 System (Re. p 43, "68" Micro Journal, May '83). Floats, etc.
F, CCF, and O - $375.00
U - $425.00
One Year Maint. - $100.00

TOLL FREE
1-800-338-6800
for Ordering
SOUTH EAST MICRO
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Reliability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

PASCAL Compilers

TSC

PASCAL Compiler

Native Code Compiler (UCSD Oriented).

F and CCF - $2[illegible].00

Lucidata

PASCAL Compiler

P-Code Compiler (ISO Standard). Designed especially for Microcomputer Systems; Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Allows Linkage to Assembler Code for maximum flexibility.

F and CCF 5" - $198.00
F 8" - $285.00

OmegaSoft

PASCAL Compiler

For the PROFESSIONAL: ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Use custom I/O devices in place of the Pascal INPUT and OUTPUT; Long Int. (32 Bit); Dynamic length strings; Interrupt processing, ROM-able, PIC, Re-Entrant Code, etc. POWERFUL! Includes Source for the Symbolic Debugger, Runtime, and several Utilities. Requires a "Motorola Compatible" Relocating Assembler and Linking Loader.

F and CCF - $425.00
One Year Maint. - $[illegible].00

DECOMPILERS

Southeast Media

DUB (A UniFLEX "basic" De-Compiler)

Re-Create a Source Listing from UniFLEX Compiled **basic** Programs. Easy to Use; works w/ ALL Versions of UniFLEX basic; Output to Disk or Terminal. Time TESTED and PROVEN: SOLID!

U - $219.95

UTILITIES

Southeast Media

Basic09 XRef

This **Basic09 Cross Reference Utility** is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also included is a **Program List Utility** which outputs the listing without the overhead of building the cross reference table, which allows it to run considerably faster when only a "pretty printed" listing with line numbers is desired. Requires Basic09 or RunB for operation.

O and CCO - Obj. Only - $75.00
O and CCO - w/ Source - $[illegible].00

Southeast Media

OS-9 VDisk

Give your OS-9 **Level** I System the **speed of memory access** that can be several orders of magnitude over your present floppy disk drive. Use that **Extended Memory** capability of your SWTPC or Gimix CPU card (or any other that has the same format DAT). The size of the Virtual Disk is completely variable in whole increments of 4K up to 960K, which is all that these systems can address beyond the base page that **OS-9 Level I** uses. By putting all of your CMDS Directory on your Virtual Disk, you can have the fastest execution speed possible (next to eating up System Memory with all of them). You can also set up high speed inter-process communications via random virtual disk files and not eat up valuable system memory with pipe buffers. Some Assembly Required - **Level I ONLY.**

O, obj. only - $79.95
w/ Source - $149.95

Southeast Media

O-P

—— OS/9 to FLEX - FLEX to OS/9 ——

Finally; the barrier has been removed between **OS/9** and **FLEX** formatted disks! Now you can READ **from**, and WRITE **to**, a Single Sided 5" or 8" FLEX diskette from OS-9 with **O-P**. **O-P** is a new and unique program, written in **BASIC09** (with Source), that performs the following functions:

COPYDISK: A **BASIC09** Program that reformats a chosen amount of an OS-9 disk to **FLEX** Format so it can be used normally by FLEX.

FLEX: A **BASIC09** Program that does the actual **read** or **write** function to the special **O-P Transfer Disk**, all selected from a user-friendly menu. Functions provided include reading the FLEX Directory, Deleting FLEX Files, Copying both directions, etc. All selections are interactive and complete, including all necessary prompts to the operator.

FLEX users can read, write and use the special disk as any other FLEX disk, provided the FLEX directory is not allowed to continue beyond track zero (too many files).

F and CCF - $79.95

Southeast Media

COPYBULK

—— Copy LARGE Disks to several smaller disks ——

The following FLEX utilities allow the backup **of ANY size disk** to any SMALLER size diskettes (Winchester to 8's or 5's, 8" to 5's, etc.). By simply inserting diskettes as requested by **COPYBULK**, a large disk system may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using the normal copy routines.

COPYBULK.CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP.CMD is a special program that downloads "random" type files, any size.

RESTORE.CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK.CMD a "bonus" utility that "relinks" the free chain of floppy or hard disk thereby eliminating fragmentation.

Completely documented source files included.

ALL 4 Programs (8" or 5") $99.95

Southeast Media

CHESS 6809

Requires FLEX and DISPLAYS **On Any Type Terminal**

Features:

*Four levels of play.
*Swap side. *Point scoring system.
*Two display boards. *Change skill level.
*Solve Checkmate problems in 1-2-3-4 moves.
*Make move and swap sides. *Play white or black.

This is one of the **strongest** CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels).

F and CCF - $79.95

VISA MasterCard

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Southeast Media

DIET-TRAC Forecaster

DIET-TRAC Forecaster is an XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P Cl) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's weight for a 30-day period is printed.

F - $59.95
U - $89.95

Southeast Media

XDATA

A **COMMUNICATION** Package
for the **UniFLEX** Operating System

Allows **UniFLEX Based Systems** to Transmit and Receive files to and from other Computer Systems via Modem. Use with CP/M, Main Frames, other UniFLEX Systems, etc.

- — Verifies Transmission Integrity using checksum or CRC
- — Automatically Re-Transmits bad blocks
- — Transmits data in 128 byte blocks

U - $299.99

Southeast Media

JUST

Text Formatter

JUST, a Text Formatter developed by Ron Anderson, provides numerous features which make it a valuable addition to any FLEX Users Software library. **JUST** is designed for formatting Text Output for Dot Matrix Printers and provides many unique features:

-Output the "Formatted" Text to the Display for format analysis and change.
-Output the "Formatted" Text to a Text File for use with the supplied **PPRINT.CMD** for producing multiple copies of the Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (this Utility is very useful at other times also, and worth the price of the program by itself).
-**User Configurable** for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); provides for up to ten (10) imbedded "Printer Control Commands", such as italics on and off, boldface on and off, etc.
-Automatic compensation for a "Double Width" printed line.
-Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.
-Use with **ANY** Editor.
-Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

F and CCF - $49.95

Lucidata

PASCAL UTILITIES

Requires LUCIDATA Pascal ver 3.

XREF — produce a Cross Reference Listing of any text; oriented to Pascal Source.

F and CCF - $25.00

INCLUDE — allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File inclusions.

F and CCF - $25.00

PRETTYLER -- produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of large programs, and provides clues as to the integrity of the program. Supplied as Source Code; requires compilation.

F and CCF - $25.00

Lucidata

COPYCAT

Pascal **NOT** required

Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directories, Copy Files, and convert Text Files when required. Also includes a Utility for investigating Physical Compatibility problems. Programs supplied in **Modular Source Code** (Assembly Language) to make it easier to solve unusual problems.

F and CCF 5" - $50.00
F 8" - $65.00

Computer Systems Consultants

FLEX DISK UTILITIES

Eighteen (18) different FLEX Utilities that should be a part of every FLEX Users Toolbox. Ten BASIC Programs to:

Compare, Merge, or **Generate Updates** between two BASIC Programs, check BASIC **Sequence Numbers**, compare two **unsequenced** files, and 5 Programs for establishing a **Master** Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files.

The other 8 Programs provide .CMD Files which allow

Copy a File with **CRC Errors**, so it can possibly be **salvaged**; **Test** Disk for errors; **Compare** two **Disks**; a fast **Disk Backup** Program; **Edit Disk Sectors**; **Linearize Free-Chain** on the Disk; print **Disk Identification**; and **Sort** and **Replace** the Disk Directory (in sorted order).

ALL Utilities **Include Source** (either BASIC or Source Code). An EXCELLENT Value!

F and CCF - $50.00

BUSINESS

Word Processors

Allard and Associates

EDITOR III

EXTREMELY Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; EXCELLENT Documentation (over 300 pages), including a full Tutorial Section to help you learn how to use the system. Features Cursor-based editing, dynamic Screen Formatting (what you see is what you get), Multi-Column display and editing, "decimal align" columns (AND add them up automatically, if wanted), define multiple keystroke macros, even and odd page number headers and footers, imbed printer control codes in text, full justification series of commands, full "help" support, store common command series on disk for future use, etc. Easy "Set-Up" (for example, you just hit the key you want to use for a specific function, such as "cursor up", and the System reads an stores that key - no digging into tech manuals for codes, etc.); use supplied "set-ups", or remap the keyboard to what you are used too. Except for proportional printing, this package will DO IT **ALL!**

6800 or 6809 FLEX or SSB DOS, OS-9 - $175.00

Great Plains Computer Co.

STYLOGRAPH

A full-screen oriented WORD PROCESSOR — (now runs on the **Data-Comp** and FHL **Color FLEX** Systems; uses the 51 x 24 Display Screens). Full screen display and editing (i.e., what you see is what you get); supports the Daisy Wheel proportional printers.

SPECIAL CCF - $195.00
F and O - $295.00
U - $395.00

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Great Plains Computer Co.

MAIL MERGE

Greatly extends the power and flexibility of **STYLOGRAPH**. Allows Multiple Text files to be printed out as one large document. Provides for merging information from the Text File during printing (such as different names and addresses), etc.

F, CCF, O - $145.00
U - $195.00

Southeast Media

SPELLS "Computer Dictionary"
OVER 120,000 words

No more "let your fingers do the walking through the Dictionary" while you are entering Text with your favorite Editor or Word Processor. **SPELLS** is more than just "another Spelling Checker"; it allows you to look up a word from within your Editor or Word Processor so that you **KNOW** it is right WHEN YOU TYPE IT IN with the **SPELL/CMD** Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View a word in context" before changing or ignoring, etc. **SPELLS** first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. **SPELLS** also allows the use of **Small Disk Storage** systems.

F and CCF - $129.95

Great Plains Computer Co.

SPELL

Fast Computer Dictionary -- allows directly changing the Text File, adding words to the dictionary, etc. 75,000 words in less than 400 sectors.

F, CCF, **OS/9** - $125.00
U - $175.00

DATA BASE MANAGEMENT SYSTEMS

Westchester Applied Business Systems

XDMS

Possibly one of the most powerful Database Management Systems' available, this machine language program is small enough to operate on a **single sided 5" disk**, yet provides the **speed** of M.I. and **power** limited only by the user's imagination. This DMS supports Relational, Sequential, Hierarchical, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. **XDMS Level I** provides a functional "entry level" System which provides for defining a Data Base, entering and changing the Data, and producing Reports. **XDMS Level II** adds the POWERFUL "**PROGRAMMING**" facility which uses an English Language Command Structure in manipulating the Data to create new File Structures, Sort, Select, Calculate, etc. **XDMS Level III** adds several special "Utilities" which provide additional ease of working with the various structures, changing System Parameters, etc.

XDMS Lvl I - F & CCF - **$129.95**
XDMS Lvl II - F & CCF - **$199.95**
XDMS Lvl III - F & CCF - **$269.95**
XDMS System Manual only - $24.95

Great Plains Computer Co.

DBFOVC DBMS

An XBASIC, Menu Driven, DBMS with "Built-In" Audit Tracking, Extremely Powerful Report & Format Capabilities, etc. This **Time Proven** DBMS will become the "Work Horse" of your Software Stable.

F and CCF $295.00
U $395.00

ACCOUNTING PACKAGES

Great Plains Computer Co.

Accounting Packages

Accts Rec., Accts Payable & Gen Ledger -- A **FULL** Accounting Package that can be used together, or as separate packages; provides the IRS required Audit Tracking. (XBASIC, based on the "Osborne Business Programs.")

F and CCF - ea. Program $295.00
U - ea. Program $395.00

Universal Data Research, Inc.

Accounting and **Database Manag. Sys.**

Note: ALL UDRI Accounting and DBM Progs. Require FLEX and **XBASIC**. These are Time Tested programs from an old, established, software house.

Color Computer FLEX Systems

DBM Part 1 - $49.95 — DBM Part 2 - $49.95
Church Contributions - $49.95
Single Entry Gen Ledger - $49.95
Balanced Billing System - $49.95
A/R $99.95 — A/P $99.95 — Gen Ledger $189.00
Inventory 2 $69.00 — Payroll $99.95

FLEX and **UniFLEX** -- Note: Requires **XBASIC** or **basic**

A/P	F - $295,	U - $395
A/R	F - $295,	U - $395
Gen Ledger	F - $295,	U - $395
Inventory 2	F - $295,	U - $395
Payroll	F - $295,	U - $395
DBM	F - $350,	U - $450

Computer Systems Consultants

FULL SCREEN INVENTORY/MRP

The Full Screen Inventory System provides a means of maintaining small inventories. Using a linked, keyed random file structure based upon the item field, it keeps the file in alphabetical order for easier inquiry. With the FIND command, the user may locate and/or print all records matching on partial or complete item, description, vendor, or attributes. Items in backorder or below minimum stock levels may be located and/or printed thru the same process. Printed output may be produced in item or vendor order. A materials requirement planning (MRP) capability for manufacturing environments is included to allow the maintenance and analysis of hierarchical assemblies of items in the inventory file. It requires **TSC's Extended BASIC**.

F and CCF - **$150.00**, U - **$150.00**

BUSINESS FORECASTING

The Virginia Company ***Bizpack***

BIZPACK is used for storing accounting, numeric, and financial data which can then be used for planning, budgeting, forecasting, analyzing, etc. While "Electronic Spreadsheets" are extremely useful in many situations, BIZPACK excels in businesses where there are numerous expense columns, revenue sources, significant business indicators, large numbers, erratic week-to-week and month-to-month fluctuations, etc. **BIZPACK** helps determine statistical relationships, establish trend lines, "smooths" data via moving averages, analyze seasonal data, adjusts for inflation, lags data in Statistics or Column functions, plots data, etc. **BIZPACK** is oriented toward time series analysis of businesses. The Program displays information on the screen in Columns of Information with each Row conforming to a defined Period of Time (weeks, months, years, etc.), and is very easy to use (data is easy to enter, change, and modify; commands can be renamed to suit the users requirements; unlimited ability to create specialized commands using common BASIC Statements; etc.). Requires **TSC's Extended BASIC**.

F and CCF - $135.00
with Source - $250.00

VISA MasterCard

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

TOLL FREE
1-800-338-6800
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F = FLEX, **CCF** = Color Computer FLEX
O = OS-9, **CCO** = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Bizpack Special

Purchase XBASIC and BIZPACK together for $221.50
— a Savings of $13.50 —

Computer Systems Consultants
TABULA RASA SPREADSHEET

TABULA RASA is similar to DESKTOP/PLAN and provides for the generation and maintenance of tabular computation schemes often used for analysis of business, sales, and economic scenarios. Its menu-driven user interface provides these capabilities even to those users with no programming experience. Its extensive report-generation capabilities allow the user to generate professional results with minimum effort. It requires TSC's Extended BASIC.

F and CCF - $100.00, U - $200.00

Computer Systems Center
DYNACALC

THE Electronic Spread Sheet for 6809 Computer Systems. An extremely POWERFUL Business Tool, this Program will find an unlimited number of "non-business" applications, also (for example, a Full Junior College Electronics Curriculum was set up using DYNACALC). Advanced features like "Table Lookup" make Income Tax work easy; Column or Row Sorting for numerous applications; etc. Completely "Memory Resident". Machine Language, this Program is FAST. PROVides STANDARD FLEX Text File output for use with BASIC, Word Processors, Pascal, "C", etc. Also available for Data-Comp and PHIL FLEX systems using the 50 x 24 Displays.

F and SPECIAL CCF - $250.00
U - $395.00

ODDS & ENDS

Computer Systems Consultants
FULL SCREEN FORMS DISPLAY

This Package supports any Serial Terminal with cursor control of Memory-Mapped Video Displays. The package substantially extends the screen Input/Output capabilities of TSC's Extended BASIC programs by providing a simple, table-driven method of describing and using full screen displays. These table entries are easy to set up and maintain, and are normally stored on disk and read as required. A simple, interactive means of generating the forms and the data field definitions is provided.

F and CCF - $50.00, U - $75.00

Computer Systems Consultants
FULL SCREEN MAILING LIST

The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Using a random file structure based on the first character of the name field, it maintains the file in alphabetical order for easier inquiry. With the FIND command, the user may locate all records matching on partial or complete name, city, state, zip, or attributes. Printed listings and output to labels may also be produced on the same selective basis. It requires TSC's Extended BASIC.

F and CCF - $150.00, U - $110.00

COLOR COMPUTER SOFTWARE

Stearns Electronics
FORTH

Intrigued by Forth??? Here is a FORTH package tailored to the Color Computer! This package is supplied on Tape, with instructions for transferring it to disk if you wish. Written primarily in machine language, it's speed is unparalleled. A full Semigraphic-8 Editor is provided, along with "goodies" like Graphics and Sound Commands, Printer Commands, Auto-Repeat and Control Keys, etc. If you are interested in Learning FORTH, a Trace Feature is provided which is invaluable. If you are a FORTH Pro, this package provides CPU carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. (Or: you won't "out grow" the Basic capabilities of this Implementation). Combine this package with Leo Brodie's EXCELLENT Book "Starting FORTH", and you will be a FORTH Expert before you know it (and have a lot of fun doing it!).

Color Computer TAPE - $98.95

Custom Software Engineering, Inc.
Color Computer GRAPHIC SCREEN PRINT Programs

Dumps any "PMODE" Screen to the Printer with the BASIC USR Function. Shift the Printout Left or Right or Reverse Print (Dark for Light Screen and Vice Versa). All Programs on Tape.

GSPR	for R.S. LP-VII/VIII & DMP 100/200/400	$7.95
GSPRE	for Epson w/ Graftrax and Graftrax +	$9.95
GSPRG	for Gemini 10 and 15	$9.95
GSPRP	for the Prowriter Printers	$9.95

Custom Software Engineering, Inc.
DATE-O-BASE CALENDAR Program

A Menu Driven EXTENDED BASIC Program which allows the entry of up to 12 Memos per Day, each of which may contain up to 28 Characters, for any day of the Month between the years 1700 and 2899. A Graphic Calendar shows which days contain Memos, and a "Key Word" Search is provided which can be output to the Screen or Printer.

TAPE DATE-O-BASE CALENDAR
(Each Tape File will hold up to 400 Memos) $16.95

DISK DATE-O-BASE CALENDAR
(4,000 Memos at 300/Month per Disk) $19.95

Custom Software Engineering, Inc.
That's INTEREST-ing

Interested in INTEREST (the Money Kind)? An EXTENDED BASIC Program that will help you deal with numerous problems requiring interest calculations. Present Value, Rate of Return, Current Bond Yield and Rate of Return to maturity, Loan Repayment Amortization Schedules, etc.

TAPE - $29.95

Custom Software Engineering, Inc.
DISK DATA HANDLER 64K

An EXTENDED BASIC Data Management System w/ Mach. Lang. Routines. Allows a max of 246 Chars. and 14 Fields per Record, and another Record can be linked to the First; 8 Char. Field Names, up to 99 Chars. per Field. Powerful On-Screen editor for Input and Update, flexible Output capabilities including output to Disk Files for use by other Programs. Change File Definition without re-entering the Data, Split Files, etc. Allows Multiple Field Sorts, Select on any combination of Fields, etc. An extremely POWERFUL TOOL; instructions provide examples of Mailing Lists and a Financial Stock Profit and Loss Tracking System.

DISK - $54.95

Custom Software Engineering, Inc.
DISK DOUBLE ENTRY

DISK EXTENDED BASIC Accounting Program w/ Mach. Lang. Routines. A "Traditional" Accounting Package for Small Business, Clubs, Churches, Personal Use, etc. Up to four levels of subtotals with Trial Balance, Income Statement, and Balance Sheet Reports. DOE allows up to 300 accounts and a Trial Balance of $9,999,999.99. Transactions may be up to 14 lines long, and comments and explanations may be freely used. Accounts are traceable to the Journal transaction, which may include comments. Screen reports allow review of past transactions and current balances.

DISK - $44.95

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Computer Systems Center

DYNASHARE

— Multi-User, Multi-Tasking with FLEX —

Southeast Media is now shipping DYNASHARE **FROM STOCK** - the multi-user, multi-tasking capability of **DYNASHARE** allows FLEX users the advantages of more sophisticated and time saving computer usage without having to buy or learn a new language or Operating System syntax. **DYNASHARE**, as its name implies, allows true "time-sharing" operation under the popular FLEX operating system, and also allows each user to run two simultaneous jobs (multi-tasking); even on single-user systems. For example, while in EDIT, you can list another file or examine a directory. Or, you might look up an item in a Data Base while a Sort is in progress! **DYNASHARE** also provides some fringe benefits that will be greatly appreciated by FLEX users, including type-ahead, command line editing, and instant response to "escape".

DYNASHARE is the painless method! Use your existing Flex computer by simply adding 64K of RAM for each user and/or task. Fact is, you still use FLEX just like you always have! **DYNASHARE** is not intended as competition to UniFLEX. It does not improve on the speed of FLEX, and does not offer password protection or other niceties of a full-blown multi-user system. What **DYNASHARE does do** is give FLEX users a **low-cost way to use existing software in a multi-user, multi-tasking environment**, so your existing FLEX versions of BASIC, XBASIC, editors, assemblers, disassemblers, sort/merge packages, word processors, compilers, DYNACALC spread-sheet package, and so on are still good.

NOTE -- The **initial release** of DYNASHARE is for **SWTPC 8/09** Computers, but versions will also be available for other popular extended-memory (up to **512K**) systems, such as HELIX and GIMIX. A minimum of **128K** of **RAM** will be required with **ALL** versions. **DYNASHARE** requires **64K of RAM for each active task:** thus a 256K system could allow foreground-background operation on two terminals, or foreground-only operation on four terminals.

AVAILABLE NOW from **Southeast Media** - **$200.00**

AUTHORS - PROGRAMMERS
QUALITY SOFTWARE NEEDED
FLEX - UniFLEX - OS/9 - Color Computer

For the past several months, we at the **Southeast Media Division** of **Computer Publishing, Inc.** (CPI), the parent company of **'68' MICRO JOURNAL** and **COLOR MICRO JOURNAL**, have debated expanding our software distribution business. Many other magazines have been doing so for years (in fact, MOST were in the Software Distribution Business BEFORE they began to publish a Magazine). Presently there are many fine examples of software that has been developed by YOU, our readers, that will never see the "light of day" due to the **Cost of Advertising** and **TIME and Cost** involved in the **production, distribution,** and **Customer SUPPORT** of that software unless SOMEONE, with enough exposure and the willingness to continually advertise, runs with the ball.

Software is the "backbone" for the REAL utilization of any Computer System, and ours are no exception! This has been no simple decision. While we realize that there could be some conflict with some of our advertisers, we ALSO hear a LOUD and CONTINOUS cry for **HELP** from our Readers. From day one, the **foremost concern** of '68' MICRO JOURNAL has been it's **READERS!** Therefore, our **Southeast Media Division** will accept, for appraisal for possible Distribution, 6809 software; Games, Utilities, Software Development, Business Application Programs, etc.

In the past there has been **too much** software offered that was not quite ready. We will strive to eliminate that element. But, right up front, we tell you only that we will do our very best; nothing more. Also, we will strive to keep cost to a bare minimum, while securing for the author a fair return in royalty payments, promptly paid, and in customer support for his product.

Of course, we will expect, no -- **DEMAND**, that the author keep the product free of errors (bugs), and maintain it in a prompt and business like manner. Also we shall require that authors be willing to furnish 'source' for those programs that justify, by price and utility, inclusion of same. The lack of source code, properly commented, is a continual complaint we hear. Not all programs will be sold with source, but where necessary, we will insist that it be included.

In some instances the program may be small or short and not justify itself as a "single" sale product. In this event it will be combined with other like programs, and offered as a package. In that event, the royalties will be split between the various authors.

If you have software that you feel will qualify under this program, please contact one of the people below. Remember, if your software has any problems or "funnies" — **GET IT STRAIGHT BEFORE YOU CONTACT US!!** Also get your source code in proper shape and well commented; there is too much 99% code already drifting around.

If your software is **READY** contact:
Bob May, Don Williams, or Tom Williams

Southeast Media is a division
of **Computer Publishing, Inc.** (CPI),
a family of 100% 68XX support facilities.

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

TEN MOST-ASKED QUESTIONS about DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. What is an electronic spread-sheet, anyway?
Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. Is DYNACALC just for accountants, then?
Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. You can build worksheets as large as 256 columns or 256 rows. There's even a built-in sort command, so you can use DYNACALC to manage small data bases — up to 256 records.

3. What will DYNACALC do for ME?
That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc™, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications. Typical uses include financial planning and budgeting, sales records, bills of material, depreciation schedules, student grade records, job costing, income tax preparation, checkbook balancing, parts inventories, and payroll. But there is no limit to what YOU can do with DYNACALC.

4. Do I have to learn computer programming?
NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Even experienced programmers can get jobs done many times faster with DYNACALC, compared to conventional programming. Built-in HELP messages are provided for quick reference to operating instructions.

5. Do I have to modify my system to use DYNACALC?
Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. Will DYNACALC read my existing data files?
You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, STYLOGRAPH™ word processor, RMS™ data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. How fast is DYNACALC?
Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. Is there a version of DYNACALC for MY system?
Probably. You need a 6809 computer (32k minimum) with FLEX™, UniFLEX™, or OS-9™ operating system. You also need a decent crt terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX and OS-9 versions of DYNACALC allow you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX (Frank Hogg or Data-Comp versions).

9. How much does DYNACALC cost?
The FLEX versions are just $200 per copy; UniFLEX version $395; OS-9 version (works with LEVEL ONE or LEVEL TWO) $250. Orders outside North America add $7 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. Where do I order DYNACALC?
See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX or OS-9 versions. Software serial number is required for the UniFLEX version.

Order your DYNACALC today!

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

CSC

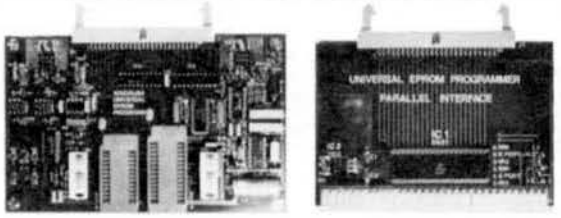
UniFLEX software prices include maintenance for the first year.

DYNACALC is a trademark of Computer Systems Center

VisiCalc is a trademark of VisiCorp.
STYLOGRAPH is a trademark of Great Plains Computer Co.
RMS is a trademark of Washington Computer Services.
FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Motorola.

WINDRUSH MICRO SYSTEMS

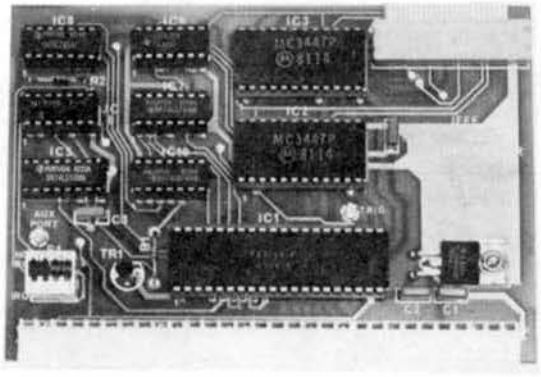
UNIVERSAL EPROM PROGRAMMER



- PROGRAMS and VERIFIES 2508, 2708, 2516, 2716, 2532, 27S32A, 2564, and 2764 EPROMS. Minor hardware mods are required to program the INTEL 27128.
- Tri-volt and Single Volt 2508/2708 and 2516/2716 devices are supported.
- ZIF sockets with mode selector switches eliminate 'personality modules'.
- Twin boards with five feet of twisted pair ribbon cable puts the programmer out on the bench where it belongs.
- SS-30 and EXORCISOR interfaces are available.
- Menu driven software provides the following facilities:
  a. MOVE blocks of memory within the buffer.
  b. READ an EPROM into the buffer.
  c. VERIFY an EPROM against the buffer.
  d. EXAMINE and change the contents of the buffer.
  e. DUMP the contents of the buffer in HEX and ASCII.
  f. FILL a selected area of the buffer with a specified character.
- Software available for all versions of SSB DOS, FLEX 2, FLEX 9 and OS-9. Assembly language source files supplied on disk....enables customising.
- Well documented users manual provides step-by-step adaptation and operating instructions.

AVAILABLE FROM GIMIX IN THE U.S.A.

IEEE-488



- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/8) BUS SPECIFICATION:
  - Talker
  - Listener
  - System Controller
  - Serial Poll
  - Parallel Poll
  - Group Trigger
  - Single or Dual Primary Address
  - Secondary Address
  - Talk only...Listen only
- Fully documented with a complete reprint of the KILOBAUD article on the IEEE bus.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. These drivers have been extensively tested and are GUARANTEED to work!
- Single SS-30 board (4, 8, or 16 addresses per port), fully socketed, gold plated bus connectors, and IEEE interface cable assembly.

PL/9

EDITOR/COMPILER/DE-BUGGER

- Friendly inter-active environment where you have INSTANT access to the Editor, the Compiler, and the Trace-Debugger, which, amongst other things, can single step the the program a SOURCE line at a time. You also have direct access to any FLEX utility and your System Monitor.
- 250 page manual is organized as a tutorial with plenty of examples.
- Fast single pass compiler produces 8K of COMPACT and FAST 6809 machine code output per minute with no run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files
- Signed and unsigned BYTEs and INTEGERs, 32-bit floating point REALs.
- Vectors (single dimension arrays) and Pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (\), negation (-)
- Expression evaluators: (=), (<>), (<), (>), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND), (.OR), (.EOR/.XOR).
- Control statements: IF..THEN..ELSE, IF..CASE1..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (CCR) and (XREG).
- FULLY supports the MC6809 SWI, SWI2, SWI3, NMI, FIRQ, IRQ and RESET vectors. Writing a self-starting (from power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library function modules are supplied: IOSUBS, DIVIO, HARDIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, and REALCON.

'... THIS IS THE MOST EFFICIENT COMPILER I HAVE FOUND TO DATE.'

Quoted from Ron Anderson's FLEX User Notes column. Need we say more?

MACE/XMACE

A co-resident EDITOR/ASSEMBLER for the 6809 written by Graham Trott which takes most of the pain out of assembly language program development:

- Friendly inter-active environment where you have INSTANT access to the Editor, the Assembler, FLEX and your System Monitor.
- MACE can also produce ASMPROC's for PL/9 with the assembly language source passed to the output file as comments.
- Includes XMACE a co-resident 6800/1/2/3/8 EDITOR/CROSS ASSEMBLER.

'C'

This is the FLEX version of the James McCosh 'C' compiler that is also available on UNIFLEX from SWTP and OS-9 from Microware:

- The FLEX implementation supports the full Kernighan and Ritchie 'C' specification except 'floats', 'doubles', and 'bit-fields'.
- Produces very efficient assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%
- Supports interleaved assembly language programs.
- The TSC relocating assembler/linking loader (SPD9-17) is REQUIRED.

MACE	(6809 FLEX only).....	$ 98.00
PL/9	(6809 FLEX only)...(a steal at this price!).....	$198.00
'C'	(a 56K system and the TSC SPD9-17 package is req'd)....	$295.00
IEEE-488	with IEEE-488 cable assembly.....	$298.00
SS-30	Universal EPROM programmer w/one version of software...	$375.00
EXORciser	Universal EPROM programmer w/one version of software...	$395.00
SOFTWARE	Drivers for a 2nd, 3rd, or 4th operating system.....	$ 25.00

ALL PRICES INCLUDE AIR MAIL POSTAGE

WE STOCK THE FOLLOWING COMPANIES PRODUCTS: GIMIX, SSB, FHL, MICROWARE, TSC, LUCIDATA, AND ALFORD & ASSOCIATES.

FLEX (tm) is a trademark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporations, MDOS (tm) and EXORciser (tm) are trademarks of Motorola Incorporated.

An SS-50C all CMOS 256K STATIC RAM board will be available SOON!

Write for details & pricing.

WORSTEAD LABORATORIES, NORTH WALSHAM, NORFOLK, ENGLAND. NR28 9SA.
TEL: (0692) 405189
TLX: 97360 SHARET G

FEATURES THE POWERFUL, THIRD GENERATION, MOTOROLA 6809 PROCESSOR!

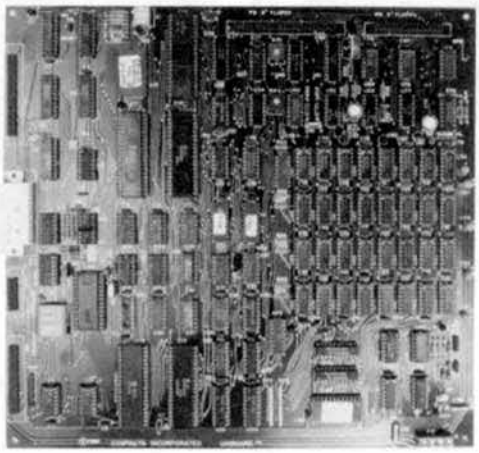
THE 6809 "UNIBOARD"™

SINGLE BOARD COMPUTER KIT

New!

PERFECT FOR COLLEGES, OEM'S, INDUSTRIAL AND SCIENTIFIC USES!

64K RAM! DOUBLE DENSITY FLOPPY DISK CONTROLLER!



BLANK PC BOARD
$99.95
WITH PAL'S, AND TWO EPROMS.

FOR 5-1/4 OR 8 INCH SOURCE DISKETTE ADD $10.

$399.00
COMPLETE KIT! FULLY SOCKETED.

ALL OPTIONS ARE STANDARD. NO EXTRAS TO BUY!

THE COMPACTA UNIBOARD™: Through special arrangement with COMPACTA INC., we are proud to have been selected the exclusive U.S. Mfg. of their new 6809 UNIBOARD™ COMPUTER KIT. Many software professionals feel that the 6809 features probably the most powerful instruction set available today on ANY 8 bit micro. Now, at last, all of that immense computing power is available at a truly unbelievably low price.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A FREE COPY IS AVAILABLE UPON REQUEST.

FEATURES:

- ★ 64K RAM using 4116 RAMS.
- ★ 6809E Motorola CPU.
- ★ Double Density Floppy Disk Controller for either 5-1/4 or 8 inch drives. Uses WD1793.
- ★ On board 80 x 24 video for a low cost console. Uses 2716 Char. Gen. Programmable Formats. Uses 6845 CRT Controller.
- ★ ASCII keyboard parallel input interface. (6522)
- ★ Serial I/O (6551) for RS232C or 20 MA loop.
- ★ Centronics compatible parallel printer interface. (6522)
- ★ Buss expansion interface with DMA channel. (6844)
- ★ Dual timer for real time clock application.
- ★ Powerful on board system monitor (2732). Features commands such as Go To, Alter, Fill, Move, Display, or Test Memory. Also Read and Write Sectors. Boot Normal, Unknown, and General Flex™.

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX™ from TSC	$149
OS9™ from Microware	$199

Specify 5-1/4 or 8 Inch

PC BOARD IS DOUBLE SIDED, PLATED THRU SOLDER MASKED, 11 x 11-1/2 IN.

Digital Research Computers
(OF TEXAS)

P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 271-3538

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order. VISA, MC, cash accepted. Add $4.00 shipping. USA AND CANADA ONLY

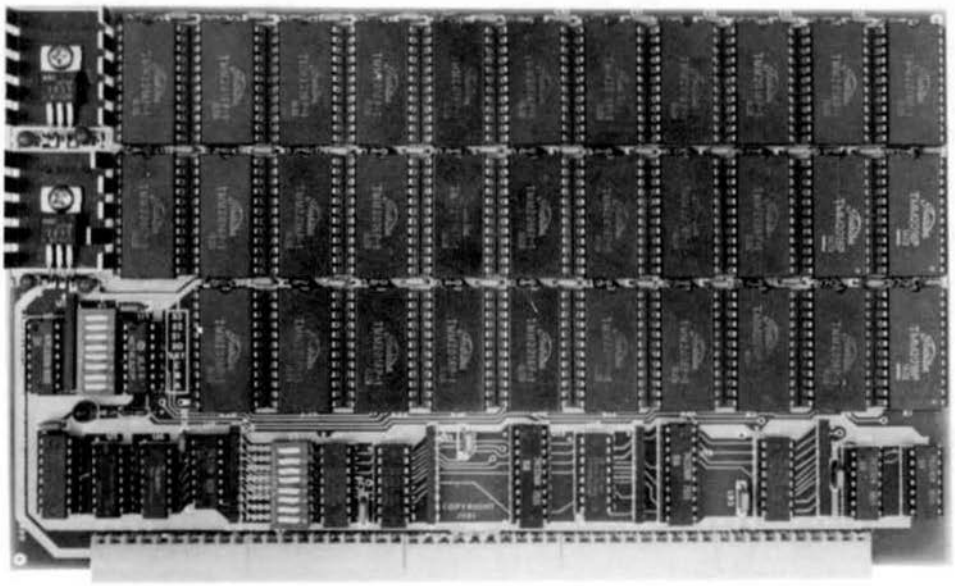
64K SS-50 STATIC RAM

NEW!

$179.00 (48K KIT)

NEW!

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $52

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	$219
64K	$249

CLOSE OUT SPECIAL
WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLD FOR $50.

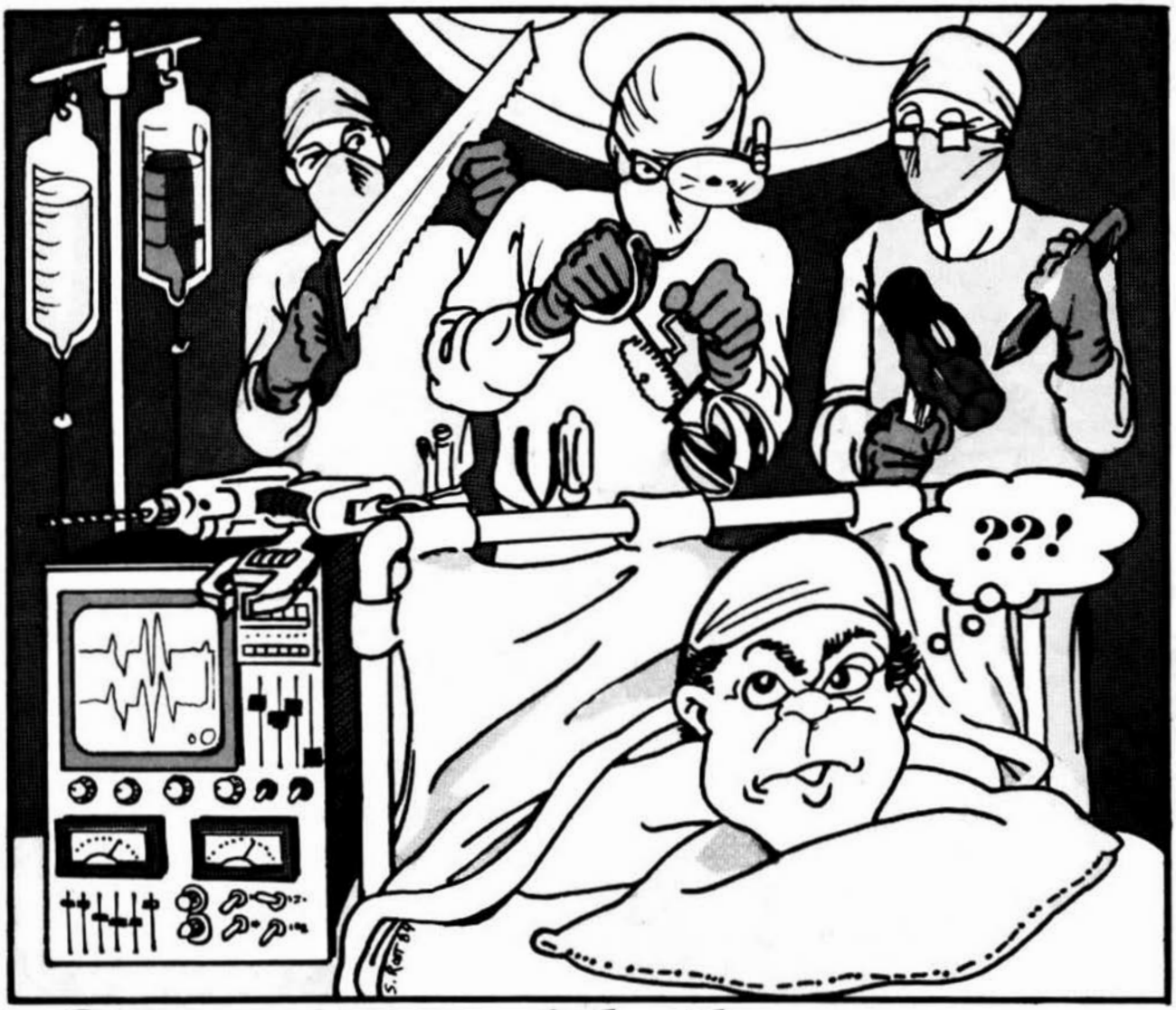
16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

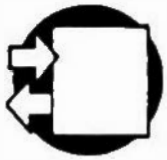
TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

SAVE THE LIFE of YOUR PROJECT!



OPERATE with the PROPER TOOLS!

JBM's MIDWARE line of quality software, is available on either five and one-quarter or eight inch diskettes. All of JBM's software packages come complete with comprehensive user's manuals.



JBM'S MIDWARE *for* OS9

OS9 is a registered trademark of Microware Corporation.

For more information, contact: Dept. 666

The JBM Group, Inc.
332 West Church Road
King of Prussia, PA USA 19406
TEL: 215-337-3138
TWX: 510-660-3999

the JBM group

VISA and MASTERCARD accepted.

68' MICRO JOURNAL ADVERTISERS INDEX

This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

GIMIX HAS THE 6809 SYSTEM TO SUIT YOUR NEEDS

HARDWARE

All systems feature the **GIMIX CLASSY CHASSIS**; with a ferro-resonant constant voltage power supply, gold plated bus connectors, and plenty of capacity for future expansion.

Static **RAM** and double-density **DMA** floppy disk controllers are used exclusively in all systems.

All systems are guaranteed for 2 MHz operation and include complete hardware and software documentation, necessary cables, filler plates, etc.

Systems are assembled using burned-in and tested boards, and all disk drives are tested and aligned by **GIMIX**.

You can add additional components to any system when ordering, or expand it in the future by adding **RAM, I/O**, etc.

GIMIX lets you choose from a wide variety of options to customize your system to your needs.

SOFTWARE

All **OS-9/FLEX** systems allow you to software select either operating system.

Also included is the **GMXBUG** monitor and, in systems with 128K or more of RAM, **GMX-VDISK** for **FLEX**.

All **GIMIX OS-9** systems include **Microware's Editor, Assembler, Debugger, Basic09,** and **Runb**; and the GMX versions of **RMS** and **DO** for **OS-9**.

All **GIMIX** versions of **OS-9** can read and write RS color computer format OS-9 disks, as well as the Microware/GIMIX standard format.

New and exclusive with **OS-9 GMX III** systems is the **GMX OS-9 Support ROM**, a monitor for **OS-9** that includes memory diagnostics and allows the system to boot directly from either hard disk or floppy.

A wide variety of languages and other software is available for use with either **OS-9** or **FLEX**.

OS-9 GMX III/FLEX SYSTEMS (#79)

The **#79** super system now includes (in addition to the above): the **GMX 6809 CPU III**, a **256K CMOS Static RAM Board (#72)**, and a **3-port Intelligent Serial I/O Processor (#11)**.

The **GMX 6809 CPU III** can perform high-speed DMA transfers from memory to memory and uses memory attributes and illegal instruction trapping to protect the system and users from program crashes. If a user program crashes, only that user is affected; other users are unaware of the problem.

The 3-Port Intelligent Serial I/O Board (#11) significantly reduces system overhead by handling routine I/O functions; freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K baud.

with dual 40 track DSDD drives	**$5998.79**
with dual 80 track DSDD drives	**$6198.79**
with **#88** dual 8" DSDD drive system	**$7698.79**
with **#90** 19MB Winchester subsystem and one 80 track	**$8898.79**
with a 47MB Winchester subsystem and one 80 track	**$10,898.79**
with a 47MB plus a 6MB removable pack Winchester subsystem and one 80 track drive	**$12,398.79**

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033.

OS-9 GMX I / FLEX SYSTEMS #49

The **#49** systems include 64KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$3998.49**
with dual 80 track DSDD drives	**$4198.49**
with **#88** dual 8" DSDD drive system	**$5698.49**
with **#90** 19MB Winchester subsystem and one 80 track	**$6898.49**

OS-9 GMX II / FLEX SYSTEMS #39

The **#39** systems include 128KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$4498.39**
with dual 80 track DSDD drives	**$4698.39**
with **#88** dual 8" DSDD drive system	**$6198.39**
with **#90** 19MB Winchester subsystem and one 80 track	**$7398.39**

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Contact GIMIX for price and availability of UniFLEX and UniFLEX GMXIII Systems.

NOTE on all drive systems: Dual 40 track drives have about 700KB of formatted capacity; dual 80's about 1,400KB; dual 8" about 2,000KB. The formatted capacity of hard disks is about 80% of the total capacity.

Want to expand your system to a megabyte of Static RAM and 15 users?

Simply add additional memory and I/O boards. Your GIMIX system can grow with your needs. Contact us for a complete list of available boards and options.

#72 256KB CMOS STATIC RAM board with battery back up	**$1898.72**
#64 64KB CMOS STATIC RAM board with battery back up	**$528.64**
#67 64KB STATIC RAM board	**$478.67**
#11 3 port intelligent serial I/O board	**$498.11**
#43 2 port serial I/O board	**$128.43**
#42 2 port parallel I/O board	**$88.42**
#95 cable sets (1 needed per port), specify board	**$24.95**

TRADE UP YOUR CoCo!

GIMIX will allow you up to $1100.00 credit toward the purchase of any GIMIX system when you trade-in your working Color Computer, peripherals, and original software. The trade-in value is limited to 110% of the RADIO SHACK™ list price at the time your order is placed. You pay the freight. This offer is good only in the Continental U.S.; is limited to the first 100 orders; and expires on 9/30/84. Only one trade-in per customer.

GIMIX INC.
1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

The Original FLEX™ for Color Computers

- Upgrade to 64K
- RS to FLEX, FLEX to RS file transfer ability
- Create your own character set
- Automatic recognition of single or double density and single or doubled sided
- All features available for either single or multiple drive systems
- Settable Disk Drive Seek Rates
- Faster High Resolution Video Display with 5 different formats
- Save RS Basic from RAM to Disk
- Move RS Basic to RAM
- Load and save function on FLEX disk
- 24 Support Commands 12 with Source Text
- External Terminal Program

DATA-COMP has everything you need to make your TRS-80C Color Computer WORK for YOU: from Parts and Pieces to Full Ready To Use Systems. DATA-COMP designs, sells, services, and SUPPORTS Computer SYSTEMS, not just Software. CALL DATA-COMP TODAY to make your Computer WORK FOR YOU!

System Requirements

FLEX9 Special General Version + Editor & Assembler (which normally sell for $50.00 ea.)	$250.00
F-MATE(RS) FLEX9 Conversion Rout. for the RS Disk Controller when purchased with Special General FLEX9 Sys.	$ 49.95
when purchased without the General FLEX9 Sys.	$ 59.95
Color Computer with 64K RAM and EXT.BASIC	$399.95

SPECIAL SYSTEM PACKAGES

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, Special General Version of FLEX9, F-MATE(RS), and a Box of 10 Double Density Diskettes, a COMPLETE ready to run SYSTEM on your Color TV Set.

$1099.95

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, and a Box of 10 Double Density Diskettes, a COMPLETE ready to run SYSTEM on your Color TV Set.

$799.95

DISK DRIVE PACKAGES, etc.

These Packages include the Radio Shack Disk Controller, Disk Drives with Power Supply and Cabinet, and Disk Drive Cable:

PAK #1 - 1 Single Sided, Double Density Sys.	$389.95
PAK #2 - 2 Single Sided, Double Density Sys.	$675.95
PAK #3 - 1 Double Sided, Double Density Sys.	$569.95
PAK #4 - 2 Double Sided, Double Denstiy Sys.	$889.95
PAK #5 - 2 Qume Thinline Double Sided, Double Density Sys	$699.95
PAK #6 - 2 Tandom Thinline Single Sided, Double Density Sys	$599.95

PRINTERS

EPSON RX-80	$375.00
EPSON MX-100	$725.00
EPSON FX-100	$650.00
EPSON FX-80	$599.00

SERIAL BOARDS

MX-Series	$119.95
RX-FX-Series	$ 99.95
Spectrum MX-Series	$ 54.95

H J L Keyboards $ 74.95

FLOPPY DISKETTES

	MEMOREX	VERBATIM
5" Soft Sector Disks		
Single Sided Single Density	$2.40ea.	$2.75ea
Single Sided Double Density	$2.40ea	$2.75ea
Double Sided Double Density		$4.92ea
Plastic Storage Box		$2.00ea
8" Soft Sector Disks		
Single Sided Single Density		$3.75ea
Single Sided Double Density		$4.18ea
Double Sided Double Density		$4.75ea
Plastic Library Box		$5.00ea

PARTS AND PIECES

Radio Shack Disk Controller	$169.95	Single Drive Disk Cable for RS Controller	$ 19.95
1 Single Sided, Double Density Disk Drive Tandom	$229.95	Double Drive Disk Cable for RS Controller	$ 24.95
1 Double Sided, Double Density Disk Drive Qume	$319.95	Micro Tech. Prods. Inc. LOWER CASE ROM Adapter	$ 74.95
1 Qume Thinline Double Sided, Double Density	$249.95	Radio Shack BASIC Version 1.1 ROM or 1.2 ROM	$ 34.95
1 Tandon 48 Track SSDD with Cabinet and Power Supply	$259.95	Radio Shack Extended Basic ROM	$ 39.95
Single Drive Cabinet with Power Supply	$ 79.95	Radio Shack Disk Basic ROM 1.1	$ 29.95
Double Drive Cabinet with Power Supply	$109.95	Screen Clean-Clears Up Video Distortion On Your Color Computer	$ 39.95
J & M Disk Controller	$144.95	Set of Eight 64K RAM Chips w Mod Instructions	$ 49.95

For Ordering Call TOLL FREE 1-800-338-6800

LAST OF PRODUCTION!!!

!! LIMITED QUANTITY !!

SWTPC 8212 & 8212(W)
Intelligent Terminals

New & Demo Models

At Discount Prices

Remaining Supply of
SWTPC 8212 CRT Terminals

EM-82 Video Terminal

Emulates the 8200 Series from SWTPC

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

DATA-COMP

P.O. Box 794 *HIXSON, TN 37343*

1-615-842-4601

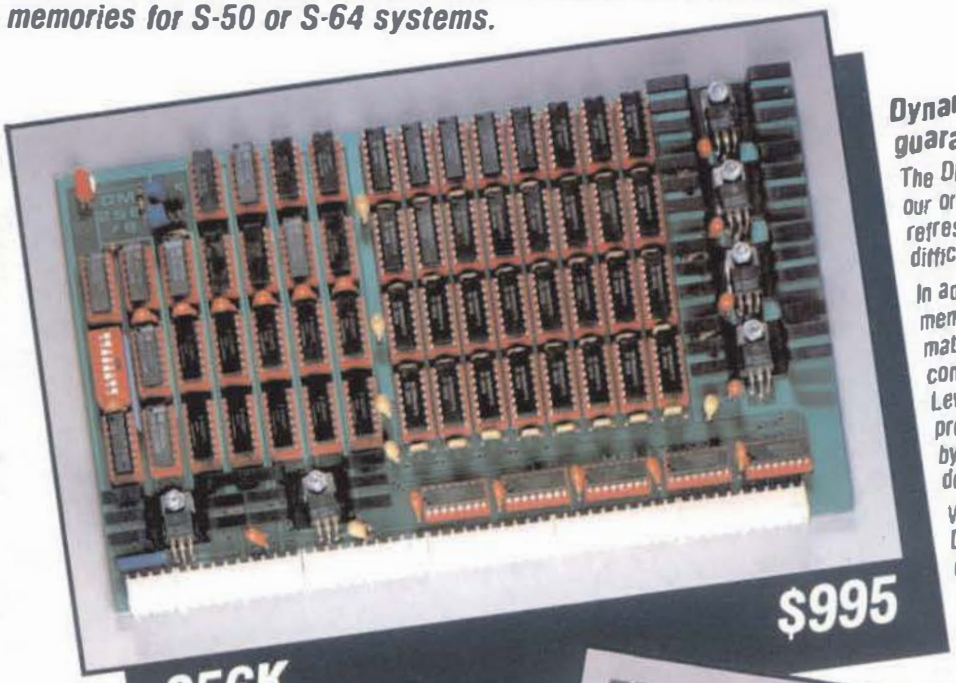
Data-Comp--South East Media & 68 Micro Journal Are Divisions of CPI

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025

MEMORY • MEMORY • MEMORY • MEM

Hazelwood Computer Systems presents a pair of advanced memories for S-50 or S-64 systems.



256K

$995

Dynamic memory with guaranteed 2.5 MHz operation

The DM-256 utilizes the same proven refresh method as our original DM-64. This technique allows fully transparent refresh while maintaining full bus speed and eliminating difficult timing problems.

In addition, individual power regulators for each row of memory ICs enhance reliability by providing the ultimate in noise decoupling. Flexible addressing mode controls make the entire 256K available to OS9 Level 2 or UniFLEX. or FLEX, the furnished utility program HYPERDSK,™ formats memory not used by FLEX as a virtual, superfast disk, thus tremendously improving total system performance.

With several hundred already in use, the DM-256 is the best choice for economical, reliable memory.

Assembled, tested, and burned in

Order: DM-256

Static memory with on-board battery backup

Utilizing the latest in low power CMOS static RAMs, the CM-64 has an on-board NiCad battery to maintain memory contents in the absence of bus power. Generous timing margins allow operation at bus speeds in excess of 2 MHz. Zener diode power loss sensing inhibits write operations to prevent data loss during power transitions. Write access can be explicitly controlled by means of an on-board jumper, a remote mounted switch, or an external logic signal.

This allows the memory to appear as ROM to the computer after the contents have been "programmed" using normal computer operations. The economical price makes the advantages of non-volatile memory available to any system.

Assembled, tested, and burned in

Order: CM-64

64K

$395

HAZELWOOD COMPUTER SYSTEMS

907 East Terra, O'Fallon, MO 63366, 314-281-1055

HELIX™

HYPERDSK is a trademark of Hazelwood Computer Systems • OS9 is a trademark of Microware • FLEX and UniFLEX are trademarks of TSC